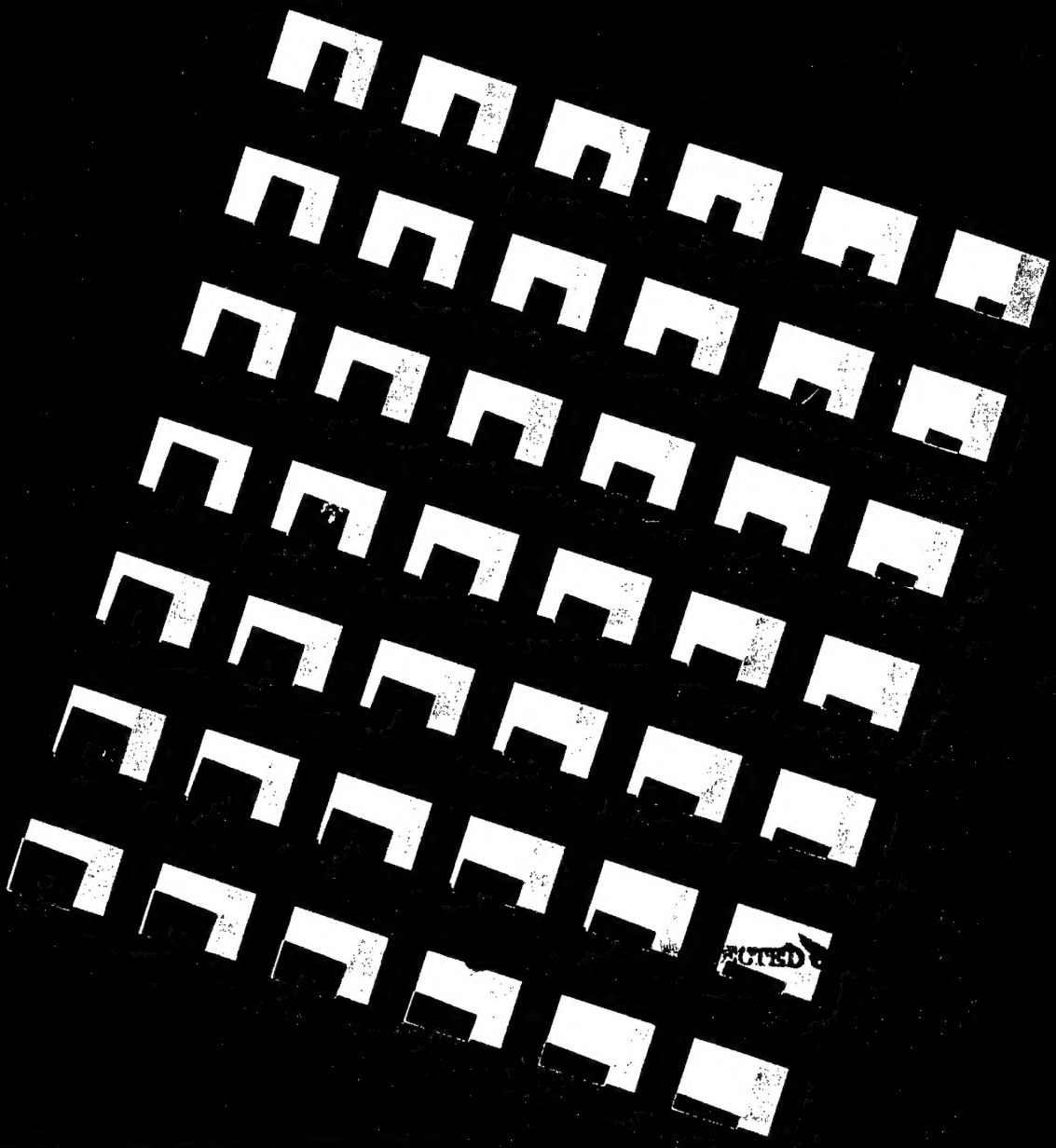
TNO-rapport
FEL-96-A307

Fast Convolution Module

TNO Fysisch en Elektronisch
Laboratorium



TNO

TNO-rapport
FEL-96-A307

Fast Convolution Module

TNO Fysisch en Elektronisch Laboratorium

Oude Waalsdorperweg 63
Postbus 96864
2509 JG 's-Gravenhage

Telefoon 070 374 00 00
Fax 070 328 09 61

Datum
oktober 1997

Auteur(s)
Dr. ir. L.H.J. Bierens
Ir. P.C.R. Beukelman
Ing. C. van 't Wout

Rubricering

Vastgesteld door	:	Ir. S.J.J. de Bruin
Vastgesteld d.d.	:	23 oktober 1997
Titel	:	Ongerubriceerd
Managementuittreksel	:	Ongerubriceerd
Samenvatting	:	Ongerubriceerd
Rapporttekst	:	Ongerubriceerd
Bijlagen A - D	:	Ongerubriceerd

Exemplaarnr.	:	9	
Oplage	:	27	
Aantal pagina's	:	67	(incl. bijlagen, excl. RDP & distributielijst)
Aantal bijlagen	:	4	

TNO Fysisch en Elektronisch Laboratorium is onderdeel van de hoofdgroep TNO Defensieonderzoek waartoe verder behoren:

TNO Prins Maurits Laboratorium
TNO Technische Menskunde

Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek TNO

Managementuittreksel

Titel	:	Fast Convolution Module
Auteur(s)	:	Dr. ir. L.H.J. Bierens, Ir. P.C.R. Beukelman, Ing. C. van 't Wout
Datum	:	oktober 1997
Opdrachtnr.	:	A93KLu777
IWP-nr.	:	771.3
Rapportnr.	:	FEL-96-A307

TNO-FEL streeft in het kader van het Synthetic Aperture Radar (SAR) programma naar onder meer de ontwikkeling en realisatie van een Real-Time SAR Processor, met als doelstelling de bruikbaarheid van real-time SAR processing voor de krijgsmacht aan te tonen.

Real-time SAR processing
SAR is bij uitstek geschikt voor het doen van verkenningen in vijandelijk gebied en het bewaken van eigen gebied met bijvoorbeeld vliegtuigen of helicopters, evt. met behulp van specifieke verkenningspods. Met de huidige resolutie is het mogelijk om o.a. voertuigen te herkennen en troepenbewegingen te volgen. In operationele situaties is het essentieel dat de SAR beelden in real-time beschikbaar zijn zodat beslissingen snel en ter plekke genomen kunnen worden.

De kritische processingstappen in een Real-Time SAR Processor zijn range compressie en azimut compressie. Range compressie is een bewerkingsstap op de ruwe SAR data die een verhoogde resolutie in de range richting oplevert. Vervolgens wordt door azimut compressie de resolutie in de vliegrichting verhoogd. Het resultaat van deze bewerkingen is het feitelijke hoge resolutie SAR beeld. Met een airborne SAR is het met de huidige stand van de technologie mogelijk om SAR beelden te genereren met een resolutie tot enkele tientallen centimeters in range en azimut richting.

Fast Convolution Module
TNO-FEL heeft een real-time SAR algoritme ontwikkeld ten behoeve van deze kritische processingstappen, waardoor het mogelijk wordt om een real-time SAR processor te realiseren. Dit algoritme is de basis van de in dit rapport gepresenteerde Fast Convolution Module (FCM). Een dergelijke module zal onderdeel gaan uitmaken van een Real-Time SAR Processor t.b.v. defensietoepassingen. De FCM zal worden ingezet om de range en/of azimut compressie in real-time uit te voeren. Met het resultaat van dit project, een geteste hardware module, is de haalbaarheid van het technologisch concept van de FCM aangetoond.

Het FCM concept wijkt af van de conventionele aanpak in het ontwikkelen van hardware t.b.v. lange convolutie- en correlatieproblemen met hoge processing

snelheden. De conventionele oplossingen gaan uit van een generieker karakter van de processing hardware, waardoor ingeleverd wordt aan processingsnelheid en -omvang. Veelal is het hierdoor niet mogelijk om SAR processing in real-time te bedrijven, zeker als beperkte omvang van de hardware een aanvullende eis is. Met de FCM is minstens een factor 5 aan reductie in gewicht, omvang en vermogen bereikt ten opzichte van de huidige, op de markt beschikbare conventionele hardware oplossingen met een vergelijkbare performance.

Toekomstige trend

In het concept van de Fast Convolution Module is impliciet rekening gehouden met het feit dat in de toekomst miniaturisatie mogelijk moet zijn. Door een tweejarige AIO (TWAIO) van de Universiteit Twente is recentelijk aangetoond dat met de huidige Aplication Specific Integrated Circuit (ASIC)-technologie de volledige functionaliteit van de FCM met behoud van de performance in één chip kan worden geïmplementeerd [2]. Momenteel wordt door TNO-FEL gewerkt aan een beschrijving van deze chip in VHDL, een hardware beschrijvingstaal t.b.v. van chip-ontwikkeling. Hiermee is een belangrijke stap gezet in de realisatie van een demonstrator Real-Time SAR Processor. Met deze demonstrator toont TNO-FEL aan dat real-time SAR processing bedreven kan worden zonder verlies aan performance of functionaliteit, met een hardware processor die aanzienlijk kleiner is en minder vermogen verbruikt dan de huidige hardware processoren.

Samenvatting

Dit rapport beschrijft het ontwerp en de realisatie van een real-time range-/azimut-compressor module, de zgn. 'Fast Convolution Module'. Deze module is gebaseerd op het fast convolution algoritme dat bij TNO-FEL is ontwikkeld. Een dergelijke module zal onderdeel gaan uitmaken van een Real-Time SAR Processor. Range en azimut compressie vormen de kritieke processingstappen van de SAR processing. Met het resultaat van dit project, een geteste hardware module, is de haalbaarheid van het technologisch concept van de Fast Convolution Module aangetoond. Deze module is getest in de Real-Time SAR Testbedomgeving van TNO-FEL. Hiermee is een belangrijke stap gezet in de realisatie van een demonstrator Real-Time SAR Processor, die aanzienlijk kleiner is en minder vermogen gebruikt dan de gangbare processoren, echter met behoud van performance èn functionaliteit.

Inhoud

Bijlagen

A	Keuze Componenten
B	Test FFT-chip
C	Globale beschrijving van de FPGAs
D	Layouts van de Printed Circuit Boards

1. Inleiding

TNO-FEL streeft in het kader van het Synthetic Aperture Radar (SAR) programma naar onder meer de ontwikkeling en realisatie van een Real-Time SAR Processor, met als doelstelling de bruikbaarheid van real-time SAR processing voor de krijgsmacht aan te tonen.

Een SAR is een beeldvormende radar met een hoog onderscheidend vermogen, onafhankelijk van de afstand. Dit maakt een SAR bij uitstek geschikt voor het doen van verkenningen in vijandelijk gebied en het bewaken van eigen gebied met bijvoorbeeld vliegtuigen of helicopters, evt. met behulp van specifieke verkenningspods. Met de huidige resolutie is het mogelijk om o.a. voertuigen te herkennen en troepenbewegingen te volgen.

In operationele situaties is het essentieel dat de SAR beelden in real-time beschikbaar zijn zodat beslissingen snel en ter plekke genomen kunnen worden. Dit vereist de aanwezigheid van een Real-Time SAR Processor met een voldoende hoge processing capaciteit aan boord van een UAV, vliegtuig of mobiel grondstation. Bovendien moet de processor compact en licht zijn en een laag vermogensverbruik hebben, zodat het in bijvoorbeeld een klein vliegtuig met verkenningstaken geïnstalleerd kan worden.

1.1 Het project

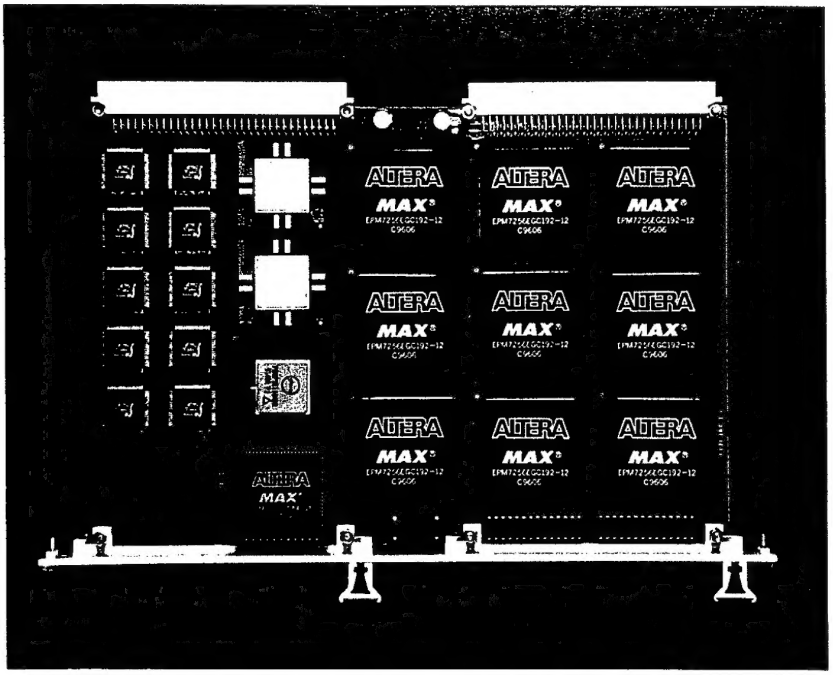
De kritische processingstappen in een Real-Time SAR Processor zijn range compressie en azimuth compressie. Range compressie is een bewerkingsstap op de ruwe SAR data die een verhoogde resolutie in de range richting oplevert. Vervolgens wordt door azimut compressie de resolutie in de vliegrichting verhoogd. Het resultaat van deze bewerkingen is het feitelijke hoge resolutie SAR beeld. Met een airborne SAR is het met de huidige stand van de technologie mogelijk om SAR beelden te genereren met een resolutie tot enkele tientallen centimeters in range en azimut richting.

Range en azimut compressie zijn beide mathematisch gezien gelijksoortige bewerkingen met een hoog rekenintensief karakter. Om de SAR processing in real-time te kunnen uitvoeren zullen de range en de azimut compressie in speciaal hiervoor ontwikkelde architecturen en hardware componenten moeten worden geïmplementeerd. TNO-FEL heeft een real-time SAR algoritme ontwikkeld ten behoeve van deze kritische processingstappen, waardoor het mogelijk wordt om een real-time SAR processor te realiseren [1]. Middels een software implementatie van het algoritme is aangetoond dat het algoritme een goede beeldkwaliteit oplevert. Op basis van de specificaties van de PHased ARray Universal SAR (PHARUS), een door TNO-FEL ontwikkeld polarimetrisch SAR systeem, is een hardware proces-

sor gesimuleerd. De resultaten hebben aangetoond dat real-time SAR processing voor PHARUS mogelijk is met een beperkte hoeveelheid hardware.

Dit real-time SAR algoritme is de basis van de in dit rapport gepresenteerde Fast Convolution Module (FCM).Een dergelijke module zal onderdeel gaan uitmaken van een Real-Time SAR Processor t.b.v. defensietoepassingen. De FCM zal worden ingezet om de range en/of azimut compressie in real-time uit te voeren. Met het resultaat van dit project, een geteste hardware module, zie Figuur 1.1, is de haalbaarheid van het technologisch concept van de FCM aangetoond.

Het FCM concept wijkt af van de conventionele aanpak in het ontwikkelen van hardware t.b.v. lange convolutie- en correlatieproblemen met hoge processing snelheden. De conventionele oplossingen gaan uit van een generieker karakter van de processing hardware, waardoor ingeleverd wordt aan processingsnelheid en processoromvang. Veelal is het hierdoor niet mogelijk om SAR processing in real-time te bedrijven, zeker als beperkte omvang van de hardware een aanvullende eis is. Met de FCM is minstens een factor 5 aan reductie in gewicht, omvang en vermogen bereikt ten opzichte van de huidige, op de markt beschikbare conventionele hardware oplossingen met een vergelijkbare performance.



Figuur 1.1: Foto van het projectresultaat: een hardware Fast Convolution Module.

1.2 Toekomstige trend

In het concept van de Fast Convolution Module is impliciet rekening gehouden met het feit dat in de toekomst miniaturisatie mogelijk moet zijn. Door een tweejarige AIO (TWAIO) van de Universiteit Twente is recentelijk aangetoond dat met de huidige Aplication Specific Integrated Circuit (ASIC)-technologie (0.8 μ CMOS) de volledige functionaliteit van de FCM met behoud van de performance in één chip kan worden geïmplementeerd [2]. Momenteel wordt door TNO-FEL gewerkt aan een beschrijving van deze chip in VHDL, een hardware beschrijvingstaal t.b.v. van chip-ontwikkeling. Hiermee is een belangrijke stap gezet in de realisatie van een demonstrator Real-Time SAR Processor. Met deze demonstrator toont TNO-FEL aan dat real-time SAR processing bedreven kan worden zonder verlies aan performance of functionaliteit, met een hardware processor die aanzienlijk kleiner is en minder vermogen verbruikt dan de huidige hardware processoren.

1.3 Opbouw van dit rapport

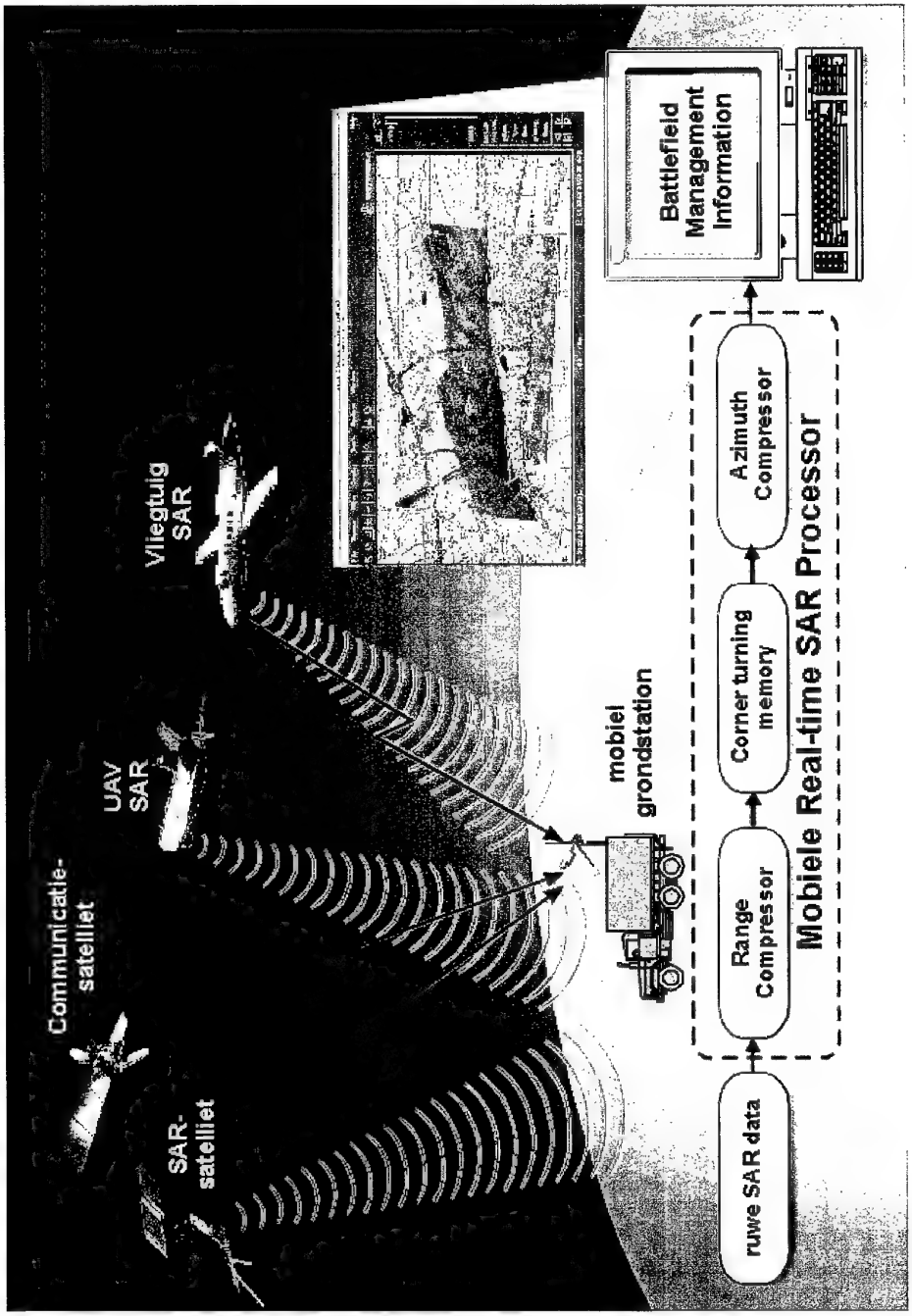
De globale opbouw van het rapport is als volgt. In hoofdstuk 2 wordt de systeemomgeving van de Fast Convolution Module beschreven en worden de ontwerpeisen vastgesteld. In hoofdstuk 3 wordt de functionaliteit beschreven door middel van toestandsbeschrijvingen. In hoofdstuk 4 wordt aan de hand van een blokschema van de module, bij elk blok de hardware ontwerpaspecten beschreven. In hoofdstuk 5 worden de belangrijkste testresultaten beschreven.

2. Systeembeschrijving

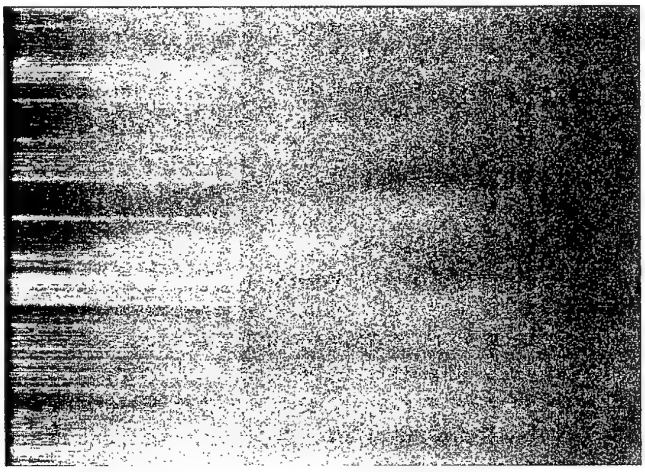
2.1 Real-Time SAR Processing

Een SAR is een side-looking imaging radar die onder een bewegend platform zoals een vliegtuig, een UAV of een satelliet is bevestigd. Gedurende een vlucht verzendt een SAR pulsen naar de aarde en ontvangt de pulsecho's. Deze pulsecho's worden gedigitaliseerd en vervolgens wordt de digitale ruwe SAR data in een SAR processingketen verwerkt tot een SAR beeld. Figuur 2.1 laat een voorbeeld van een volledige SAR keten zien. De SAR processing kan bijvoorbeeld in real-time worden uitgevoerd in een mobiel grondstation of in het radar platform zelf. In dit laatste geval kan het SAR beeld naar het grondstation worden verzonden en vervolgens, eventueel in combinatie met beelden afkomstig van andere sensoren, direct worden gedisplayed.

De SAR processingketen bevat een aantal kritische processingstappen, namelijk range compressie, corner turning en azimut compressie. Range compressie is de correlatie van de pulsecho's met een replica van de uitgezonden puls, waardoor de resolutie in de range richting verbeterd wordt. Vervolgens wordt de data set getransformeerd, d.w.z. de data wordt in een zogenaamd corner turning memory "kolom"-gewijs ingelezen en "rij"-gewijs uitgelezen. Vervolgens wordt de azimut compressie uitgevoerd, d.w.z. dat de resolutie van de SAR data in de vliegrichting wordt verhoogd. Hierbij wordt gebruikt gemaakt van een synthetische apertuur, die wordt opgebouwd m.b.v. de Doppler informatie geïnduceerd door de vliegsnelheid van het platform. Figuur 2.2 illustreert de SAR processingketen aan de hand van live SAR data van de PHased ARray Universal SAR (PHARUS), een door TNO-FEL ontwikkeld polarimetrisch SAR systeem.



Figuur 2.1: Toepassing van real-time SAR processing in een mobiel grondstation.



(a)

(b)

(c)

Figuur 2.2: SAR keten: ruwe SAR data (a), SAR data na range compressie (b) en SAR data na azimut compressie, het feitelijke SAR beeld (c).

2.2 Real-Time SAR Systeemspecificaties

De Fast Convolution Module (FCM) is een hardware implementatie van het fast convolution algoritme zoals beschreven in [1]. De functie van de FCM is het convolueren van twee discrete signalen bestaande uit complexe samples. De FCM zal worden gebruikt om de range- en/of azimutcompressie uit te voeren ten behoeve van real-time SAR processing.

Met het beoogde resultaat zal het volgende worden aangetoond:

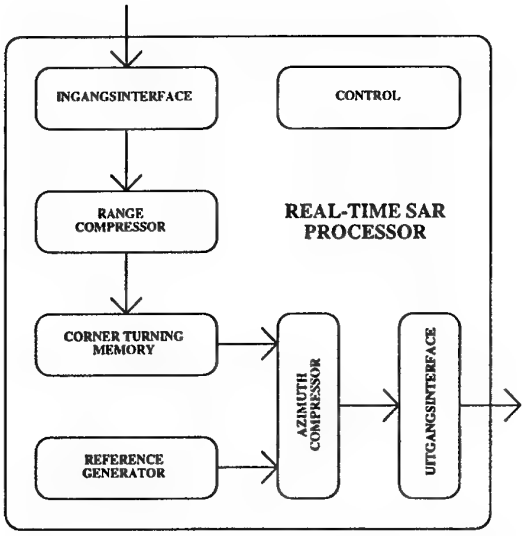
- de haalbaarheid van het technologisch concept van het fast convolution algoritme
- de mogelijkheid van real-time SAR processing met een beperkte hoeveelheid hardware
- toepassingen voor de KLu.

De systeemspecificaties van de FCM zijn gebaseerd op de specificaties van de Real-Time SAR Processor, die op TNO-FEL wordt ontwikkeld. De FCM zal dan ook getest worden in de Real-Time SAR Testbedomgeving, welke is beschreven in [3]. De systeemspecificaties van de FCM zullen hoge resolutie real-time SAR in de toekomst mogelijk moeten maken.

Figuur 2.3 toont een schematische weergave van een real-time SAR processor. De ingangsdata bestaat uit ruwe SAR data en replica data die door de PHARUS radar interface wordt aangeleverd. De ingangsinterface zorgt ervoor dat deze data op de juiste wijze in de processor wordt gelezen.De belangrijkste functies van de processor zijn de range- en azimutcompressie en de corner turning memory. Range- en azimutcompressie bestaan beide uit dezelfde basisoperaties. Er is daarom gekozen om de input/output specificaties van beide gelijk te houden. De communicatie-interfaces zorgen ervoor dat de processor onafhankelijk van de gekozen range- en azimutcompressie hardware gespecificeerd kan worden. De range en azimut compressie zal worden uitgevoerd door de Fast Convolution Module.

De corner turning memory is zodanig gespecificeerd dat zijn ingangsformaat overeenkomt met het uitgangsformaat van de compressiefuncties. In Figuur 2.3 zijn alleen de functies weergeven die minimaal nodig zijn om real-time SAR processing te bedrijven. Het SAR beeld zal met deze configuratie zgn. 'quick-look' kwaliteit bevatten, d.w.z. de resolutie is nog beperkt (ca. 5 m). Een uiteindelijke hoge resolutie Real-Time SAR Processor zal worden uitgebreid met een bewegingscompensatiemodule (om gefocusseerde beelden te garanderen bij resoluties van minder dan 2 m) en een multilook processor (om 'speckle ruis' te reduceren).

De instructies worden in standaardformaat aan de desbetreffende functionele blokken geleverd.



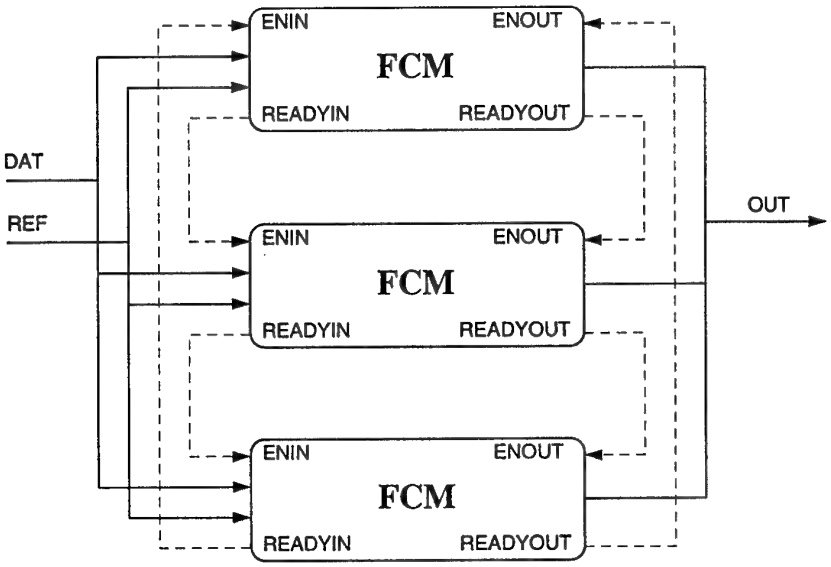
Figuur 2.3: Systeemopzet Real-Time SAR Processor.

2.3 Input/Output Specificaties

De input/output specificaties van de FCM zijn dezelfde als de input/output specificaties van de range- en azimutcompressor van de processor. In Figuur 2.4 zijn de in- en uitgangssignalen weergegeven. In Figuur 2.7 zijn de timing van de in- en uitgangssignalen getoond. De datarate van de FCM (kloksnelheid) is vastgelegd op 20 MHz. Dit betekent dat op deze snelheid de data- en referentiereeksen in- en uitgelezen zouden moeten worden. Om de flexibiliteit van de in- en uitlees datarate te garanderen, wordt dit ondervangen door aan de ingang en uitgang van de FCM buffers te plaatsen die zorgen voor de overgang van de (over het algemeen veel lagere) in- en uitgangs datarate naar de 20 MHz. De configuratie is zodanig dat een constante effectieve datarate (aantal ingangssamples gedeeld door de verwerkingstijd) van 2 MHz gewaarborgd wordt.

Om de effectieve datarate te verhogen wordt de mogelijkheid geboden om meerdere FCMs parallel te zetten. De in- en uitgang van de FCMs kunnen elkaar aansturen d.m.v. de signalen ENIN, ENOUT, READYIN en READYOUT, zie Figuur 2.4. Als een FCM de ingangsdata heeft ingelezen wordt de ingang uitgezet en wordt een READYIN puls gegenereerd. Deze wordt aangesloten op de ENIN van de volgende FCM, die daarop zijn ingang open zet (d.w.z. data van de ingangsbus kan lezen). Als een FCM de uitgangsdata heeft uitgelezen wordt de uitgang uitgezet en wordt een READYOUT puls gegenereerd. Deze wordt aangesloten op de ENOUT van de volgende FCM, die daarop zijn uitgang open zet (d.w.z. zijn uitgangsdata op de uitgangsbus kan schrijven). Bij initialisatie zal de eerste FCM zijn in- en uitgang

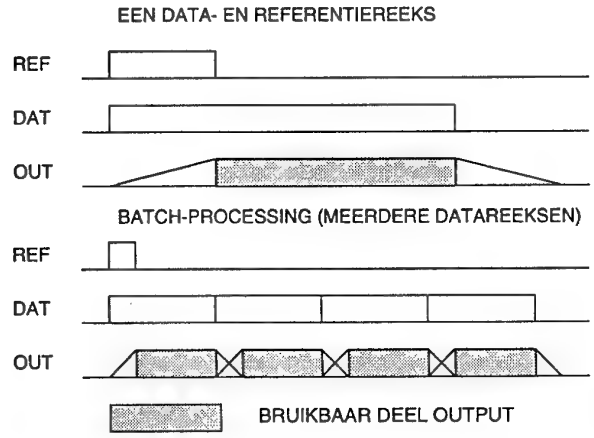
beide open zetten. D.m.v. een signaal FIRSTFCM kan worden aangegeven welke FCM de eerste is.



Figuur 2.4: Parallel schakelen van FCMs.

De data- en referentiereeks (DAT, EXPD en REF) worden parallel ingelezen. Het in- en uitgangssampleformaat is 2×16 bits two's complement integers, en een 7 bits two's complement exponent voor de data. De EXPD is een blokexponent, d.w.z. elk sample binnen een datareeks heeft dezelfde exponent.

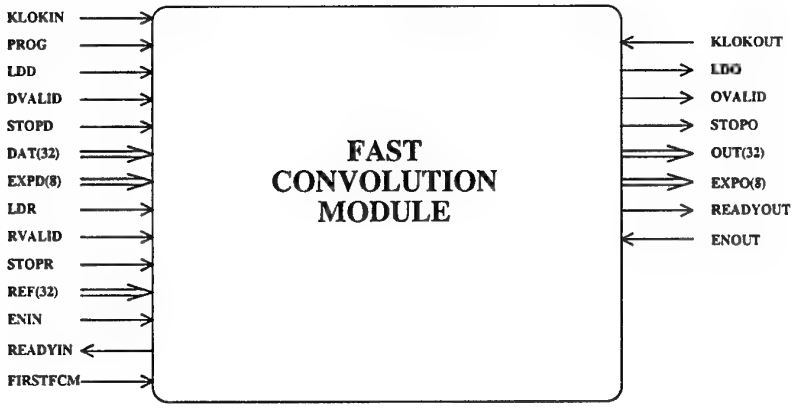
Er wordt ervan uitgegaan dat de FCM een datareeks en referentiereeks met een gezamenlijke lengte van 32 Ksamples aangeboden krijgt. Dit is een gevolg van de dimensionering m.b.t. de beschikbare hardware componenten.



Figuur 2.5: Input en output reeksen met en zonder batch-processing.

Aangezien voor de toepassingen van de FCM een datareekslengte van max. 5 Ksamples en referentiereekslengte max. 2 Ksamples is vereist, wordt de maximale processingcapaciteit bij lange na niet benut. Om de efficiëntie te verhogen is daarom gekozen voor de mogelijkheid van *batch processing*. De FCM kan meerdere datareeksen achter elkaar inlezen en in een processing slag (als ware het een lange datareeks) convolueren met een referentiereeks. De volledige output reeks bevat dan een batch bestaande uit kortere output reeksen. Een voorbeeld is gegeven in Figuur 2.5. Hierbij geldt dat per datareeks een blokexponent wordt ingelezen. De lengte van de datareeksen moet in het geval van *batch processing* een veelvoud van de halve FFT-lengte zijn (in het geval van de FCM is dit 128). Hierdoor wordt de uiteindelijke hardware implementatie ten aanzien van de exponentafhandeling sterk vereenvoudigd.

De uitgangsreeks (OUT en EXPO) kan op twee manieren worden uitgelezen: met inloop en zonder inloop. In het algemeen wil men bij convolutie de complete reeks hebben: inloop, bruikbaar deel en uitloop. Bij *batch processing* worden alleen de bruikbare delen van de outputreeksen uitgelezen. De EXPO is, evenals EXPD, een blokexponent die geldt voor de hele batch.



Figuur 2.6: Input/output signalen van de FCM.

Het begin van een reeks (zowel aan de ingang als uitgang) wordt aangekondigd door een start-signaal (LDD, LDR en LDO). Een valid-signaal geeft aan of de reeks geldig is (DVALID, RVALID en OVALID). Een stop-signaal geeft het einde van de reeks aan (STOPD, STOPR en STOPO).

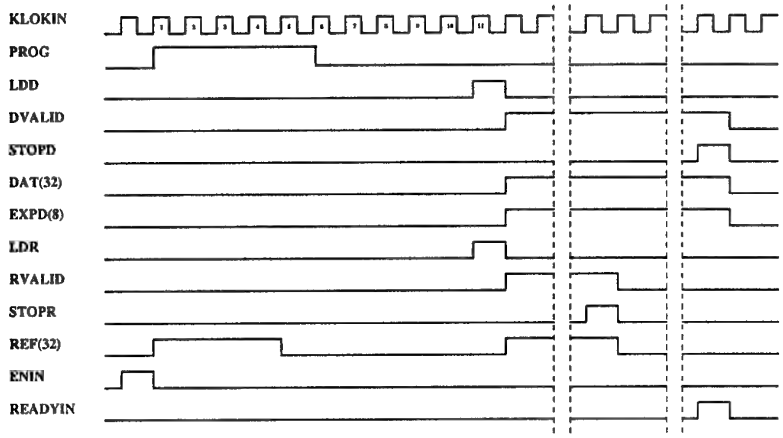
Aan het begin van een nieuwe reeks worden instructies van de referentiebus gelezen. Voor elke instructie is een byte gereserveerd. Een prog-signaal (PROG) geeft aan dat programmeerwoorden van de referentiebus gelezen kunnen worden. Een reeks instructiebytes kan niet langer duren dan 10 klokslagen.

De instructies bevatten o.a. de volgende informatie:

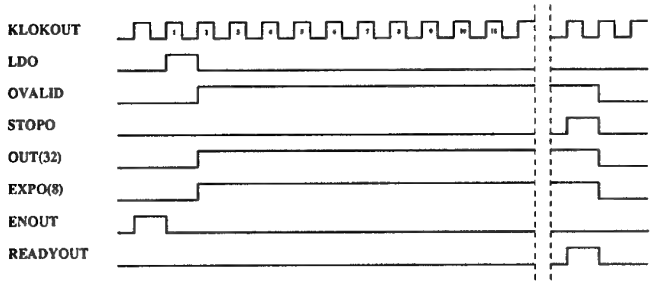
- data- en referentiereekslengte
- aantal datareeksen per batch
- outputreekslengte
- wel of geen inloop van de outputreeks
- hergebruik referentie
- FFT processor modes.

De informatie komt van een hoger gelegen control processor binnen.

De in- en uitgang kunnen apart worden geklokt d.m.v. KLOKIN en KLOKOUT. De maximale dataklokfrequentie is 20 MHz.



(a)



(b)

Figuur 2.7: Timing ingangssignalen (a) en timing uitgangssignalen (b).

2.4 Haalbaarheidseisen

Ten aanzien van het uiteindelijke hardware ontwerp van de FCM gelden de volgende eisen:

1. het gebruik van Commercial-off-The-Shelf (COTS) componenten en programmeerbare logica
2. een beperkte omvang (zo mogelijk een (1) PCB, euro 6)
3. een gemiddelde datarate van minstens 2 MHz (het aantal data-ingangssamples gedeeld door de processingtijd)
4. een vermogensverbruik van minder dan 25 W
5. inpassing in de Real-Time SAR Testbedomgeving.

2.5 Systeemtest Specificatie

Het uiteindelijke systeem zal allereerst getest worden met eenvoudige data- en referentiereeksen die een voorspelbaar uitgangsresultaat opleveren zodat het functioneren geverifieerd kan worden. Bij de keuze van de testreeksen zal rekening gehouden worden met een aantal factoren:

a. convolutieresultaat, door twee blokken op de ingangen te zetten (reëel, imaginair, combinatie hiervan)
b. elektrisch gedrag, door reeksen met veel overgangen (pulstreinen) op de ingangen te zetten
c. esthetisch gedrag, door twee chirps op de ingang te zetten die bij convolutie een sinc op moeten leveren
d. batch gedrag, wel/geen aanloop

Uiteindelijk zal het systeem gedemonstreerd worden aan de hand van een aantal radarecho's die met de gespecificeerde PRF en datarate zullen worden aangeboden. Hiertoe zal echte radardata worden gebruikt.

2.6 Ontwerpstrategie

De FCM is opgesplitst in hiërarchische functies. Van elke functie is de input/output relatie en de werking beschreven. Data en control flows en datastructuren zijn gespecificeerd en op consistentie gecontroleerd. Voor zover mogelijk is de configuratie parametriseerbaar gehouden. Voor het functioneel ontwerpen is gebruikgemaakt van de ontwerptool Teamwork. Teamwork maakt gebruik van de Structured Analysis Method. Bovendien kunnen control functies als state-machines worden beschreven. Het functioneel ontwerp is zoveel mogelijk gesimuleerd.

Het architectuur ontwerp is uit het functionele ontwerp gegenereerd. Hierbij is de keuze van de parameters bepaald. Deze parameters bepalen vervolgens de benodigde hardware configuratie. De hardware configuratie is tot poortniveau uitgewerkt in het detailontwerp. Hiervoor zijn de CAE ontwerp- en simulatietools (o.m. Dazix, Cadnetics, Altera) beschikbaar op TNO-FEL. Tijdens het testen en simuleren van het architectuur- en detailontwerp werd de timing geverifieerd en het gedrag gesimuleerd met echte SAR data.

3. Functionele Beschrijving

3.1 Algemene Beschrijving

De FCM voert het fast convolution algoritme uit in een aantal stappen. Deze stappen zijn afgeleid van het algoritme aangevuld met in- en uitlezen. In totaal worden er zes stappen onderscheiden:

1. Inlezen van referentiereeks in het werkgeheugen en kolom FFT van de data.
2. Rij FFT van de resultaten van 1.
3. Kolom FFT van de referentie.
4. Rij FFT van de resultaten van 3.
5. Vermenigvuldigen van data met referentie en daarna rij IFFT van het resultaat.
6. Kolom IFFT van resultaten van 5 uitlezen van het werkgeheugen en overlap-add operatie.

De kolom en rij FFT/IFFT duiden op de wijze waarop de data- of referentiesecties uit het geheugen worden gelezen. Hiervoor wordt verwezen naar de paragraaf 3.4.

Het uitvoeren van een stap gebeurt door de FCM in een bepaalde toestand ('state'). Als de FCM zich in een state bevindt, kan er slechts een stap worden uitgevoerd. Daarnaast is er een idle state gedefinieerd, die de rusttoestand aangeeft. Elke state heeft het nummer van de stap die wordt uitgevoerd in deze state. De idle state heeft nummer 0. In sommige gevallen is het handig om meerdere datareeksen met 1 referentie te processen. In dit geval kan processingtijd bespaard worden door states 3 en 4 over te slaan. Het toestandsverloop kan als volgt worden weergegeven:

- nieuwe data- en referentiereeks:
  state 0 -> state 1 -> state 2 -> state 3 -> state 4 -> state 5 -> state 6 -> state 0
- alleen nieuwe datareeks:
  state 0 -> state 1 -> state 2 -> state 5 -> state 6 -> state 0

Als de FCM deze toestanden doorlopen heeft is een convolutie uitgevoerd van de ingangsdata met de ingangsreferentie.

Het is mogelijk om de resultaten met en zonder convolutie-aanloop uit te lezen. Ook is het mogelijk om een aantal datareeksen in een batch te processen. Er wordt dan een convolutie uitgevoerd zonder de aanloop en de uitloop mee te nemen en de resultaten van de afzonderlijke reeksen worden door de FCM aan de uitgang gescheiden m.b.v. extra start/stop signalen. De lengte van de datareeksen moet een veelvoud van de halve FFT-lengte zijn. Deze randvoorwaarde vereenvoudigt de exponentafhandeling.

Om er voor te zorgen dat het op lage snelheid aanbieden van data de prestaties van de FCM niet naar beneden haalt, worden aan de in- en uitgangs-FIFO buffers geplaatst. De data wordt langzaam in de ingangsbuffer geschreven en daarna snel uitgelezen. Voor de blokexponent wordt een klein FIFO genomen, omdat per reeks slechts 1 exponent wordt aangeboden. Bij het uitgangsbuffer wordt de data snel ingelezen en langzaam uitgelezen.

3.2 Context Fast Convolution Module

De context van de FCM op functioneel nivo is weergegeven in Figuur 3.1. De FCM wordt voorzien van data- en programmeersignalen. Programmeerinformatie wordt via het imaginaire referentie-ingang aangeboden. Een korte beschrijving van de in- en uitgangssignalen met tussen haakjes hun breedte in bits volgt hieronder:

input:

PROG (1)	geeft aan of er programmeerinformatie wordt aangeboden op de imaginaire referentie ingangsbus
START (1)	geeft aan wanneer de module start met het in het geheugen lezen van de volgende data- en referentiereeks
LDD (1)	geeft aan dat het eerste data sample eraan komt
STOPD (1)	geeft aan dat het laatste data sample wordt ingenomen
DVALID (1)	geeft aan dat er geldige samples op de databus staan
RVALID (1)	geeft aan dat er geldige samples op de referentiebus staan
LDR (1)	geeft aan dat het eerste referentie sample eraan komt
STOPR (1)	geeft aan dat het laatste referentie sample wordt aangeboden
DAT (32)	complexe datareeks (I/Q 2*16 two's complement)
EXPD (7)	exponent van data (binaire positieve exponent)
REF (32)	complexe referentiereeks (I/Q 2*16 two's complement)
KLOKIN (1)	klok waarop alle ingangsdata en control synchroon wordt aangeboden.
KLOKOUT (1)	klok waarop alle uitgangsdata en control synchroon wordt afgegeven.
RESET (1)	reset van de FCM
ENIN (1)	dit signaal wordt 1 puls actief om de data inname mogelijk te maken. Het wordt gebruikt als er meerdere FCM kaarten samenwerken. Een vorige kaart geeft hiermee de data inname over. Bij gebruik van 1 kaart moet ENIN altijd actief zijn.
READYIN (1)	dit signaal wordt 1 KLOKIN cyclus actief nadat de data inname voltooid is. Het geeft hiermee de inname van data door aan een volgende FCM kaart.
FIRSTFCM (1)	is 1 als de betreffende FCM als eerste actief wordt na een RESET, en anders 0.

output:

OUT (32)	complexe outputreeks (I/Q 2*16 two's complement)
EXPO (7)	exponent van output (binaire positieve exponent)
STARTO (1)	geeft aan wanneer de module start met afgeven van een resultaten reeks
LDO (1)	geeft aan dat het eerste resultaat sample eraan komt
STOPO (1)	geeft aan dat het laatste resultaat sample van een (sub)reeks wordt afgegeven
OVALID (1)	geeft aan of er geldige datasamples op de uitgangsreeks staan
ENOUT (1)	wordt een KLOKOUT cycles actief om aan te geven dat deze FCM kaart mag beginnen met data afgifte. Bij gebruik van 1 FCM kaart moet ENOUT altijd actief worden gemaakt.
READYOUT (1)	dit signaal wordt 1 KLOKOUT cyclus actief nadat de data afgifte voltooid is. Het geeft hiermee de afgifte van data door aan een volgende FCM kaart.

Voorafgaand aan de data wordt programmeerinformatie aangeboden. Tijdens een actief PROG signaal haalt de FCM programmeerinformatie van de imaginaire referentiebus. De FCM verlangt hierbij de volgende informatie:

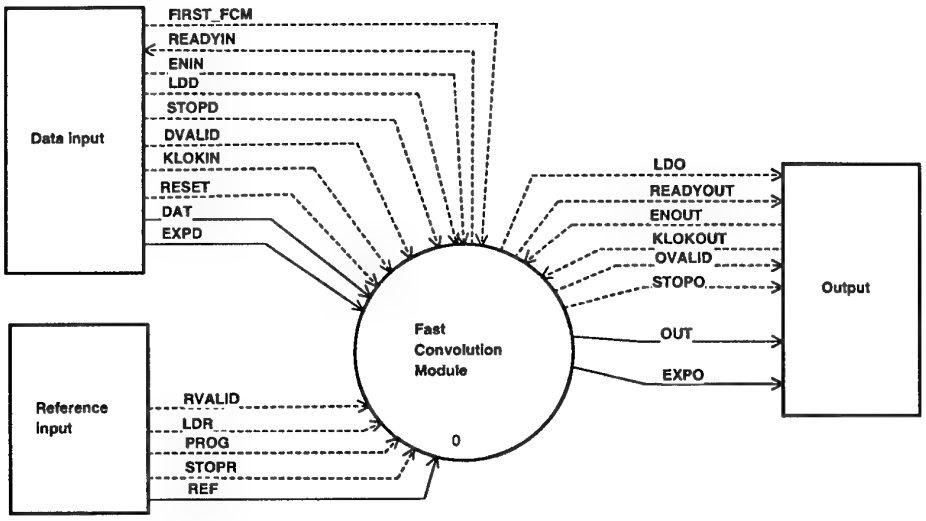
- een vlag voor herhaald gebruik van de referentie
- een vlag die bepaalt of er wel of geen aanloop wordt uitgelezen
- de lengte van de datareeks min 1
- de lengte van de referentiereeks min 1
- de lengte van de outputreeks min 2
- instellingen voor de FFT-chip
- het aantal datareeksen per batch.

Samenvattend zijn er 3 modes:

1. convolutie met aanloop
2. convolutie zonder aanloop
3. batch convolutie zonder aan- en uitloop.

Bij alle drie de modes kunnen zowel de datasectie als de referentiesectie worden vernieuwd, of alleen de datasectie. De benodigde gegevens zijn:

L_REF lengte van de referentie	(15 bits, max 32768-L_DAT)
L_DAT lengte van de data (per reeks)	(15 bits, max 32768-L_REF)
L_OUT lengte van de output (per reeks)	(15 bits, max 32768)
N_DAT aantal datasecties per batch	(8 bits)
F_AANLOOP met of zonder aanloop	(1 bit)
F_REF herhaald gebruik referentie	(1 bit)
2_3SHIFT shift mode van de FFT processor	(1 bit)



Figuur 3.1: Context diagram van de Fast Convolution Module.

3.3 Programmeerinformatie

Om de FCM op de juiste manier te laten functioneren moeten er bij de besturing een aantal parameters aanwezig zijn. Deze parameters worden overgedragen m.b.v. programmeerwoorden. De programmering gaat vooraf aan de data op de imaginaire referentiebus. Een aparte flag (PROG) geeft aan of er programmeergegevens op de referentiebus staan. De programmeerwoorden worden niet zoals de referentiesamples in een ingangs-FIFO geschreven, maar direct door de besturing opgenomen en is onafhankelijk van de state van de FCM. De programmeerinformatie is getoond in Figuur 3.2.

sample 1:

F_AANLOOP	L_DAT(14:0)

sample 2:

X	L_REF(14:0)

sample 3:

x(15:10)	2_3SHIFT	F_REF	N_DAT(7:0)

sample 4:

X	L_OUT(14:0)

sample 5:

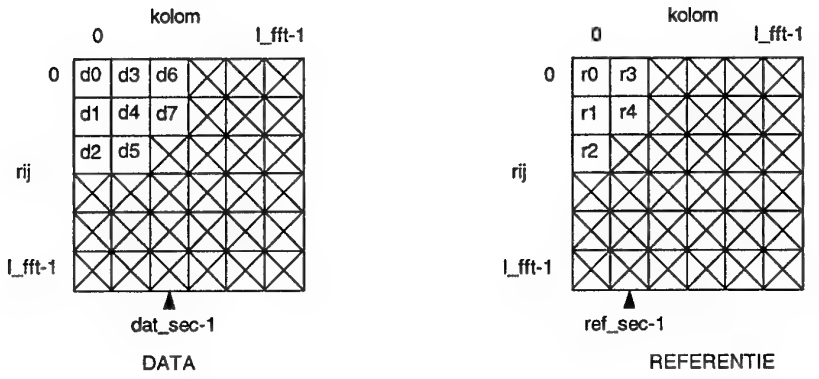
x(15:8)	EXP_G(7:0)

Figuur 3.2: De beschikbare programmeerinformatie.

3.4 State Beschrijving

Hieronder volgt een beschrijving van de verschillende states met start en eindcondities. Bij de beschreven activiteiten wordt het inlezen en afgeven van data buiten beschouwing gelaten omdat dit gebeurt m.b.v. in- en uitgangs-FIFO's. Hierdoor wordt in- en uitlezen onafhankelijk van de state waarin de FCM zich bevindt.

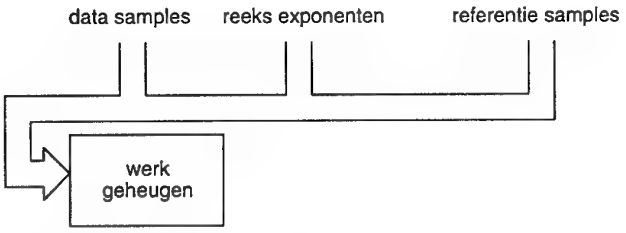
In Figuur 3.3 worden de parameters L_FFT (=FFT reekslengte), DAT_SEC =aantal datasecties met lengte L_FFT/2 binnen een datareeks), en REF_SEC (=aantal referentiesecties met lengte L_FFT/2 binnen een referentiereeks) in relatie met het geheugenbeheer verduidelijkt. De data- en referentiegeheugens worden voorgesteld als (L_FFT X L_FFT) matrices. De bepaling DAT_SEC en REF_SEC wordt beschreven in de state beschrijving van state 1. Het geheugenbeheer per state wordt verduidelijkt in Figuur 3.10.



Figuur 3.3: Verschillende parameters in het memorymanagement. Ter illustratie is gekozen voor een datareekslengte en referentiereekslengte van 8 respectievelijk 5 samples, en een 6 punts FFT. Dit impliceert dus een sectionering in secties bestaande uit 3 samples, 3 datasecties en 2 referentiesecties.

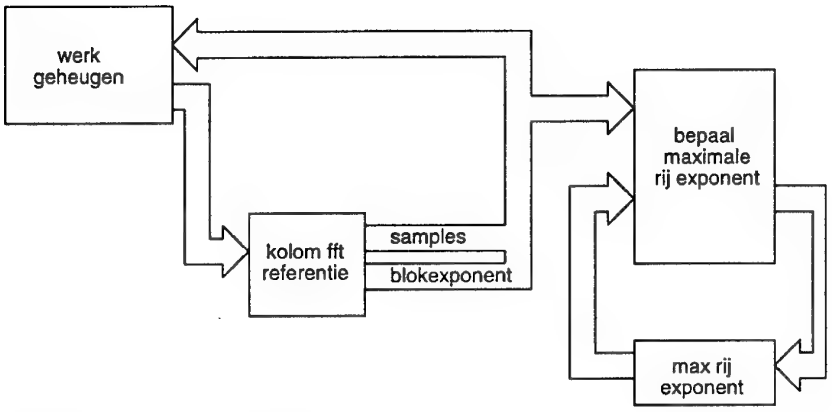
State 0:	Idle
in-voorwaarde:	Na RESET of na state 6.
beschrijving:	Rusttoestand van de FCM. Geheugeninhoud blijft gehandhaafd.
exponentafhandeling:	Geen.
uit-voorwaarde:	Bij STOPD van de laatste datareeks in een batch.

State 1:	Inlezen van referentiereeks in het werkgeheugen en kolom FFT van de data.
in-voorwaarde:	Na state 0.
beschrijving:	N_DAT datareeksen worden uit de ingangs-FIFO gelezen in secties met lengte L_FFT/2 aangevuld met nullen tot L_FFT en vervolgens getransformeerd (FFT). De resultaten van deze kolom FFT worden daarna kolomsgewijs in het datageheugen geschreven. DAT_SEC (aantal halve kolommen gevuld met data, zie Figuur 3.3) wordt bepaald door een teller. Als F_REF geldt, wordt de oude referentiereeks opnieuw gebruikt. Als F_REF niet geldt, wordt de referentiereeks in secties met lengte L_FFT/2 uit de referentie-ingangs-FIFO gelezen en kolomsgewijs in het referentiegeheugen geschreven. REF_SEC (aantal halve kolommen gevuld met referentie, zie Figuur 3.3) wordt bepaald door een teller.
exponentafhandeling:	De blokexponenten van de getransformeerde datareeksen worden gevormd door de exponent van de inkomende data samples op te tellen bij de exponent van de transformatie. Deze exponent wordt bij de data samples in het datageheugen geschreven. De maximale exponent van een sample wordt bepaald aan de hand van het meest significante bit in het sample en de blokexponent van de kolom. Bij het terugschrijven van de eerste kolom worden de maximale exponenten apart opgeslagen als rij-exponenten in het maximale rij exponent buffer. Bij het terugschrijven van elke volgende kolom wordt deze rij-exponent geupdated als de maximale exponent groter is dan de rij-exponent.
uit-voorwaarde:	Als de laatste data-kolom getransformeerd en teruggeschreven is.



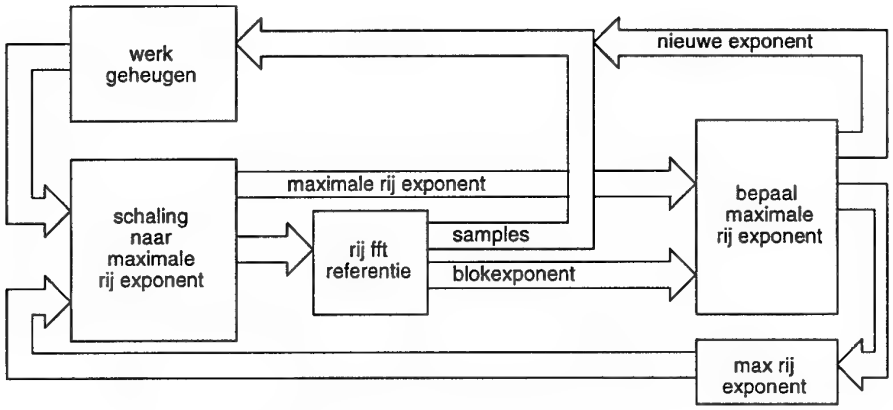
Figuur 3.4: Data flow in state 1.

State 2:	Rij FFT data
in-voorwaarde:	Na state 1.
beschrijving:	De L_FFT rijen met lengte DAT_SEC worden uit het datageheugen gelezen en aangevuld met nullen tot lengte L_FFT. Elke rij wordt getransformeerd en rijgewijs teruggeschreven in het geheugen.
exponentafhandeling:	De rijen worden voor transformatie naar de rij-exponenten geschaald. Deze rijexponent wordt opgeteld bij de transformatie-exponent en de som wordt bij de data-samples in het geheugen geschreven.
uit-voorwaarde:	Als alle L_FFT rij FFT's zijn uitgevoerd.



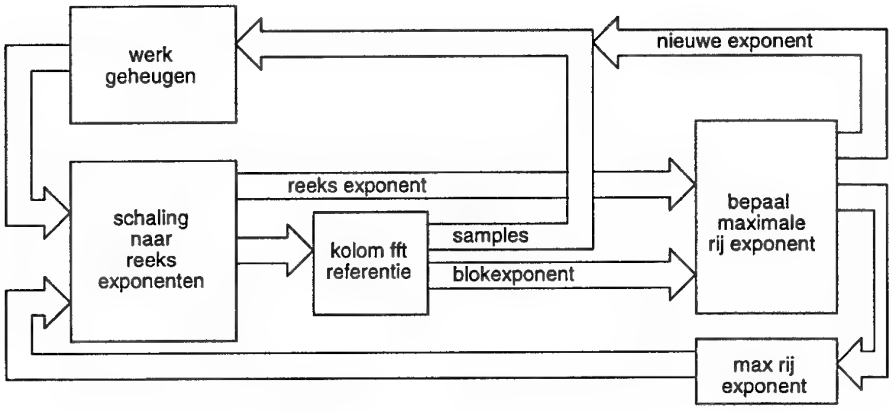
Figuur 3.5: Dataflow in state 2.

State 3:	Kolom FFT referentie
in-voorwaarde:	Na state 2 als F_REF geldt.
beschrijving:	De REF_SEC halve kolommen worden zonder hun exponenten uit het referentiegeheugen gelezen en aangevuld met nullen tot lengte L_FFT. Elke kolom wordt getransformeerd en kolomsgewijs teruggeschreven in het geheugen.
exponentafhandeling:	De blokexponenten van de getransformeerde referentiereeksen worden gevormd door de exponent van de transformatie. Deze exponent wordt bij de referentiesamples in het referentiegeheugen geschreven. De maximale exponent van een sample wordt bepaald aan de hand van het meest significante bit en de blokexponent van de kolom. Bij het terugschrijven van de eerste kolom worden de maximale exponenten opgeslagen als rij-exponenten. Bij het terugschrijven van elke volgende kolom worden deze rij-exponenten geupdated als de maximale exponent groter is dan de rij-exponent.
uit-voorwaarde:	Als alle REF_SEC kolom FFT's zijn uitgevoerd.



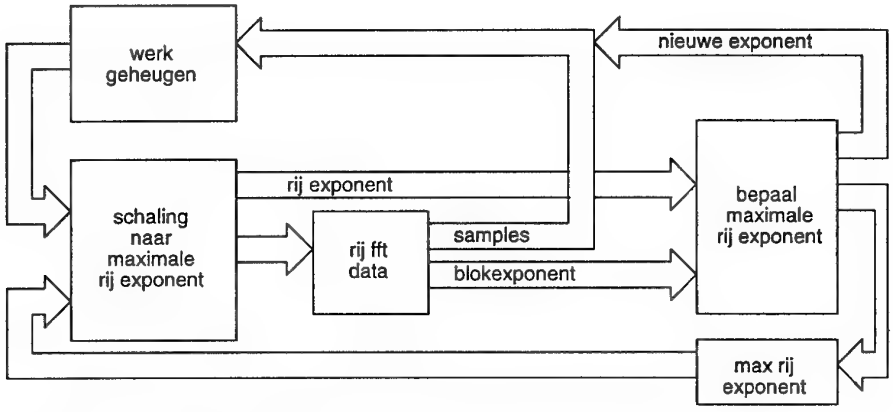
Figuur 3.6: Dataflow in state 3.

State 4:	Rij FFT referentie
in-voorwaarde:	Na state 3.
beschrijving:	De L_FFT rijen met lengte REF_SEC worden uit het referentiegeheugen gelezen en aangevuld met nullen tot lengte L_FFT. Elke rij wordt getransformeerd en rijgewijs teruggeschreven in het geheugen.
exponentafhandeling:	De rijen worden voor transformatie naar de rijexponenten geschaald. Deze rijexponent wordt opgeteld bij de transformatie exponent en de som wordt bij de referentie samples in het geheugen geschreven.
uit-voorwaarde:	Als alle L_FFT rij FFT's zijn uitgevoerd.



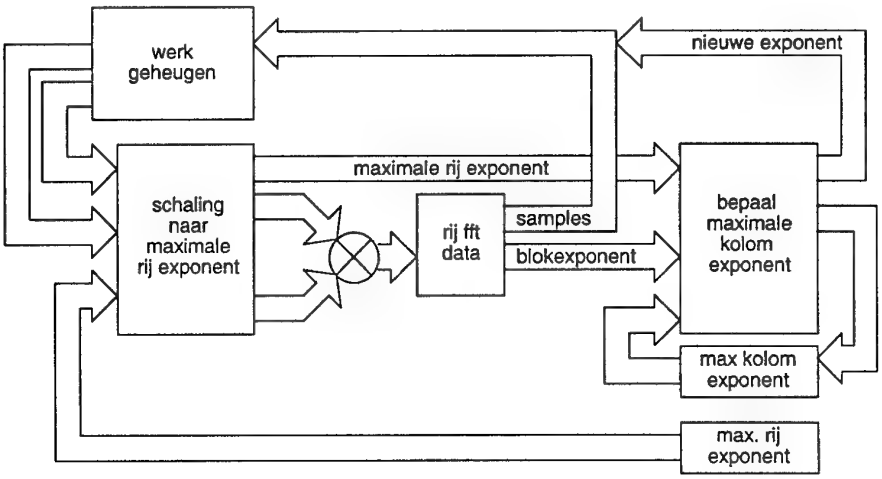
Figuur 3.7: Dataflow in state 4.

State 5:	Vermenigvuldig data- en referentierijen en IFFT resultaat
in-voorwaarde:	Na state 4 als F_REF niet geldt of na state 2 als F_REF geldt.
beschrijving:	De L_FFT rijen met lengte L_FFT worden parallel uit het data- en referentiegeheugen gelezen. Elke rij wordt elementsgewijs vermenigvuldigd en teruggetransformeerd. Het resultaat wordt rijgewijs teruggeschreven in het datageheugen.
exponentafhandeling:	De blokexponenten van de getransformeerde (IFFT) reeksen worden gevormd door de som van de twee exponenten voor vermenigvuldiging, de exponent van de transformatie en de exponent van de multiplier. Deze exponent wordt bij de datasamples in het datageheugen geschreven. De maximale exponent van een sample wordt bepaald aan de hand van het meest significante bit en de blokexponent van de rij. Bij het terugschrijven van de eerste rij worden de maximale exponenten opgeslagen als kolomexponenten. Bij het terugschrijven van elke volgende rij worden deze kolom-exponenten geupdated als de maximale exponent groter is dan de kolomexponent.
uit-voorwaarde:	Als alle L_FFT rij IFFT's zijn uitgevoerd.



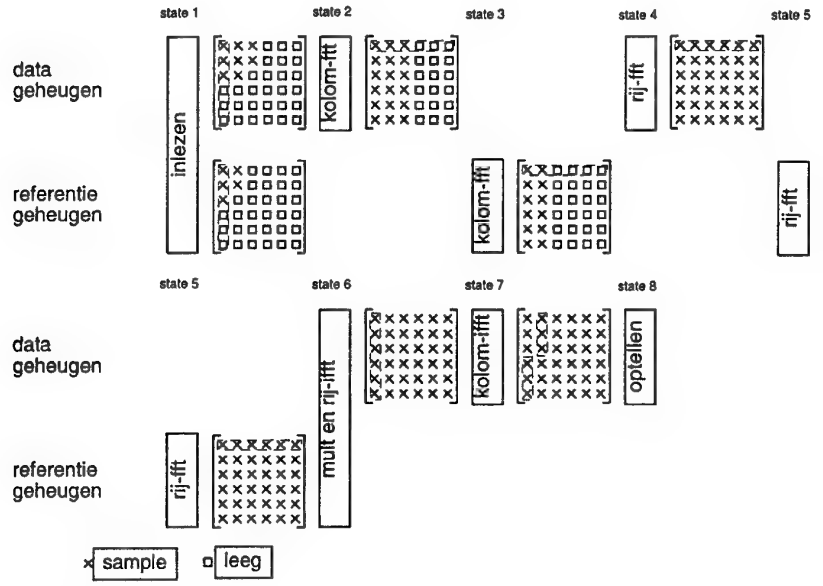
Figuur 3.8: Dataflow in state 5.

State 6:	Kolom IFFT en overlap-add
in-voorwaarde:	Na state 5.
beschrijving:	De samples worden kolomsgewijs uit het data geheugen gelezen en getransformeerd (IFFT). Van het resultaat van elke kolom worden de eerste L_FFT/2 samples naar een opteller gestuurd en de tweede L_FFT/2 samples naar het data geheugen. De andere ingang van de opteller wordt tijdens het arriveren van de eerste L_FFT/2 samples voorzien van de laatste L_FFT/2 samples van de vorige reeks uit het data geheugen. Omdat er bij het arriveren van de allereerste reeks nog geen resultaten van een vorige reeks zijn, worden bij de eerste helft van deze reeks nullen opgeteld. Bij laatste L_FFT/2 samples van de allerlaatste reeks worden eveneens nullen opgeteld.
exponentafhandeling:	De kolommen worden voor transformatie geschaald naar de kolom-exponenten. De blokexponenten van de getransformeerde (IFFT) reeksen worden gevormd door de som van de exponenten voor IFFT en de exponent van de transformatie. Deze exponent wordt naar de adder gestuurd (eerste L_FFT /2 samples) of bij de datasamples in het datageheugen geschreven (tweede L_FFT/2 samples). Van alle berekende blokexponenten in deze state wordt de grootste onthouden om tijdens het afgeven van data naar buiten de samples naar te schalen. I.v.m. de optel operatie wordt bij deze grootste blokexponent nog 1 opgeteld.
uit-voorwaarde:	Als alle L_FFT kolom IFFT's zijn uitgevoerd en de tweede helft van de laatste kolom uit het datageheugen is gelezen.



Figuur 3.9: Dataflow in state 6.

Format:	Formatting
beschrijving:	Formatting van de resultaatsamples vindt plaats nadat de FCM van state 6 naar state 0 is overgegaan. Het is geen state op zichzelf. Het moet klaar zijn voordat de FCM opnieuw in state 6 is aangekomen.
uit-voorwaarde:	Bij N_DAT=1 & geldige F_AANLOOP: Als er L_OUT (=L_DAT+L_REF-1) samples uit het uitgangs-FIFO gelezen en geschaald zijn. Bij N_DAT=1 & niet geldige F_AANLOOP: Nadat er F_REF-1 samples uit het uitgangs-FIFO gelezen zijn zonder dat deze op de uitgangsbus gezet zijn en L_OUT (=L_DAT) samples uit het uitgangs-FIFO gelezen en geschaald zijn die wel op de uitgangsbus komen. Bij N_DAT>1: Er worden afwisselend F_REF-1 samples uitgelezen en weggegooid en L_OUT(=L_DAT-L_REF+1) samples uitgelezen en geschaald. Als dit N_DAT maal gebeurd is, kan er gestopt worden met het uitlezen van de FIFO.

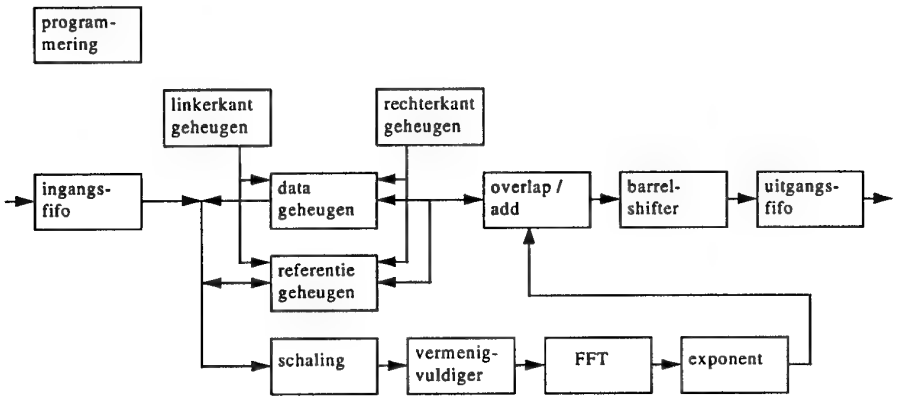


Figuur 3.10: Geheugenbehandeling per state. Data- en referentiegeheugen worden gerepresenteerd als (6 X 6) matrices. Ter illustratie is gekozen voor een datareekslengte en referentiereekslengte van 8 respectievelijk 5 samples, en een 6 punts FFT. Dit impliceert dus een sectionering in secties bestaande uit 3 samples, 3 datasecties en 2 referentiesecties. Elke matrix in de figuur representeert het respectievelijke data- of referentiegeheugen 'nadat' de bewerking gedurende een state heeft plaatsgevonden. Bovendien wordt d.m.v. balken in de matrices aangegeven op welke 'wijze' (d.w.z. koloms- of rijgewijs of als half versprongen kolommen) de geheugenelementen beschikbaar zijn voor de volgende bewerking. De lege geheugenplaatsen worden bij het uitlezen vervangen door nullen (zero-padding).

4. Hardware Detailontwerp

4.1 Algemeen

Voor de hardware implementatie is zoveel mogelijk gebruik gemaakt van off-the-shelf componenten die de te verrichten functionaliteit al in zich hebben. De functionaliteiten die niet (efficiënt) in bestaande componenten kunnen worden geïmplementeerd, alsmede de besturing van de bestaande componenten, zijn geïmplementeerd in programmeerbare logica (FPGA's). De opbouw van de schakeling is in Figuur 4.1 blokschematisch weergegeven. De schakeling wordt per functie behandeld.[1]



Figuur 4.1: Blokschema van de Fast Convolution Module.

4.2 Data Flow

Verdeling van data over meerdere printkaarten

Om de aangeboden data te kunnen verdelen over meerdere printkaarten zijn er twee mechanismen geïmplementeerd. Het eerste mechanisme zoekt naar een FCM die niet bezet is en laat deze FCM aan de dataverwerking beginnen. Het tweede mechanisme zorgt ervoor dat de gecorreleerde reeksen van opeenvolgende FCM's elkaar netjes opvolgen.

---

1 In dit hoofdstuk zal de naamgeving van de signalen uit de vorige hoofdstukken worden gehandhaafd. In bijlage C is een conversielijst opgesteld voor de naamgevingen zoals ze in de FPGA source code zijn gebruikt. De overige signalen die nog niet geïntroduceerd zijn in de vorige hoofdstukken, zijn signalen die in de FPGAs zijn gebruikt, zie bijlage C.

Aanwijzing eerst beschikbare FCM

Het mechanisme dat zoekt naar een FCM die nieuwe data kan gaan innemen werkt als volgt. Zodra de spanning opkomt of als er een reset gegeven wordt zal elke FCM kijken naar z'n FIRSTFCM ingang. Slechts één van de FCM's heeft een 1 op deze ingang staan en mag daarom de allereerste reeks innemen. Alle andere FCM's hebben een 0 op de FIRSTFCM ingang staan en doen daarom voorlopig nog niets. Zodra de eerste FCM voldoende data ingenomen heeft zal deze een puls afgeven op z'n READYIN uitgang. Dit wordt ontvangen op de ENIN ingang van de volgende FCM, wat voor deze FCM een sein is om de eerstvolgende reeks in te gaan nemen. Indien er maar één FCM wordt toegepast dan moeten FIRSTFCM en ENIN continu 1 gemaakt worden.

Beschikbaar stellen van het correlatieresultaat.

Het mechanisme dat de uitgangsresultaten van de verschillende FCM's netjes aan elkaar knoopt werkt op een soortgelijke manier. De FCM die middels het FIRSTFCM signaal aangewezen wordt zal de data gelijk op de uitgangsbus zetten zodra er gecorreleerde data beschikbaar is. Zodra alle data afgegeven is zal het signaal READYOUT geactiveerd worden. Dit wordt op de ingang ENOUT van de volgende FCM ontvangen, wat voor deze FCM een sein is om data op de uitgangsbus te zetten zodra deze beschikbaar is. Indien er maar één FCM wordt toegepast dan moet de ingang ENOUT continu geactiveerd worden.

Toestand van de totale schakeling

De FCM verwerkt de binnenkomende data in verschillende stappen voordat het resultaat beschikbaar is. De verschillende stappen worden weergegeven door state-machine *fcm_state*.[2] Deze state-machine is in één FPGA geïmplementeerd en wordt vervolgens doorgegeven naar alle andere FPGA's. Door deze centrale oplossing lopen alle FPGA's altijd met elkaar in de pas.

Klokken

De data wordt bij de FCM aangeboden op de opgaande flank van de dataklok KLOKIN. De data wordt door de FCM echter verwerkt op 40MHz of op 20MHz. De 40MHz klok wordt op de kaart gegenereerd en wordt vervolgens gedeeld naar 20MHz. Omdat veel componenten van 20MHz moeten worden voorzien komt het 20MHz signaal op 5 pinnen beschikbaar. Om alle klokken schoon te houden wordt elk kloksignaal aan het einde van het printspoor afgesloten met een weerstand van 2K26.

---

2 De state machines die in dit hoofdstuk worden genoemd, zijn geïmplementeerd in de FPGA's, zie bijlage C.

Op verschillende plaatsen bevinden zich overgangen van de ene klok naar de andere klok. Vooral de overgang van KLOKIN (waarvan de frequentie onbekend is) naar de 20MHz klok is een lastige waaraan extra aandacht is besteed door middel van handshake signalen (zie PROG_BEEN en OVAD_OUT).

4.3 Programmering

De FCM kent een aantal instellingen die net voor het begin van een datareeks worden overgedragen. Zodra het signaal PROG geactiveerd wordt zal de data op de imaginaire referentiebus geïnterpreteerd worden als programmeerdata. Afhankelijk van de volgorde en de positie van deze data krijgt het een betekenis als programmeerwoord (zie paragraaf 3.3). De programmeerwoorden worden binnengehaald zodra het PROG signaal geactiveerd is maar worden pas daadwerkelijk van kracht zodra de FCM weer aan een nieuwe verwerkingscyclus begint. Dit wordt aangegeven door state-machine *fcm_state*, deze moet zich dan in de toestand READ_PROG bevinden. Zodra alle programmeerwoorden ingelezen zijn gaat de state machine over naar de volgende toestand.

4.4 Ingangs-FIFO

Indien een FCM aan de beurt is om data te gaan verwerken (wat bepaald wordt met het mechanisme zoals in paragraaf 2.3 en 4.2 is beschreven) zal het signaal ENIN_LONG actief zijn. De aangeboden data wordt dan opgevangen in drie FIFO's. Eén voor de data, één voor de referentie en één voor de exponent. De data wordt in de ingangs-FIFO geschreven zolang DVALID actief is. De signalen LDD en STOPD hadden hier ook voor gebruikt kunnen worden maar daar is geen gebruik van gemaakt omdat DVALID het meest universeel is. Indien dat in de toekomst veranderd moet worden dan kan dat eenvoudig omdat ze wel worden aangeboden bij de FPGA's. De exponent wordt in de ingangs-FIFO geschreven tijdens de eerste klokcyclus dat DVALID actief is. De referentie wordt in de ingangs-FIFO geschreven indien er aan een aantal voorwaarden wordt voldaan. Deze voorwaarden zijn:

- De FCM moet aan de beurt zijn om data in te nemen (ENIN_LONG is actief).
- Alleen de eerste referentie reeks wordt ingenomen (wat van belang is tijdens batch mode).
- Het RVALID signaal moet actief zijn.
- Middels een programmeerwoord moet aangegeven worden dat de referentie verwerkt moet worden.

De signalen LDR en STOPR hadden in plaats van RVALID gebruikt kunnen worden om de referentie in te nemen. Dit is echter niet gedaan maar zou in de toekomst eenvoudig veranderd kunnen worden omdat deze signalen wel worden aangeboden bij de FPGA's.

Om te bepalen of alle data ingelezen is wordt het aantal ingelezen reeksen geteld (SEQ_CNT). Indien het aantal ingelezen reeksen gelijk is aan het aantal reeksen dat volgens het programmeerwoord N_DAT moet worden ingenomen, dan wordt gestart met het uitlezen van de FIFO's.

De data die uit de ingangs-FIFO wordt gehaald, wordt naar het geheugen of naar de schaling overgebracht. De overdracht wordt gecoördineerd vanuit de logica die ook zorg draagt voor het besturen van de linkerkant van het geheugen. Het lezen van de FIFO is daarom ook beschreven in paragraaf 4.5. Zodra alle data uit de ingangs-FIFO's gelezen is, worden ze gereset. Dit houdt in dat er geen nieuwe reeksen in de FIFO's kunnen worden geschreven als de voorgaande reeksen nog niet uit de FIFO gelezen zijn.

4.5 Linkerkant van het Geheugen

Tussen de ingangs-FIFO, de linkerkant van het geheugen en de schaling bevindt zich een databus. De besturing van de datastromen over deze bus gebeurt centraal. Om de besturing overzichtelijk te houden is deze hiërarchisch opgezet. Bovenin de hiërarchie zit de state-machine *left*. Deze bepaalt de richting van het dataverkeer over de bus. Als tweede in de hiërarchie zit de state-machine *lijn*. Deze begeleidt het versturen van een lijn van 256 samples. Als derde in de hiërarchie zitten tellers om het einde van een lijn, een kolom of een veld te bepalen en logica om de besturingssignalen naar de ingangs-FIFO, de linkerkant van het geheugen en de schaling te verzorgen. De globale datatransport acties die plaatsvinden zullen in de volgende paragraaf worden behandeld.

4.6 Schaling

In de FCM worden acht bewerkingen uitgevoerd op een data veld van 256 bij 256 punten. Elk data punt bestaat uit een mantisse en een exponent. Omdat de bewerkingen 'FFT' en 'multiply' alleen kunnen worden uitgevoerd op een getal zonder exponent moet de data naar de meest optimale exponent geschaald worden waarna de mantisse en de exponent een eigen weg door het datapad gaan volgen. In het blok 'schaling' worden deze acties ondernomen.

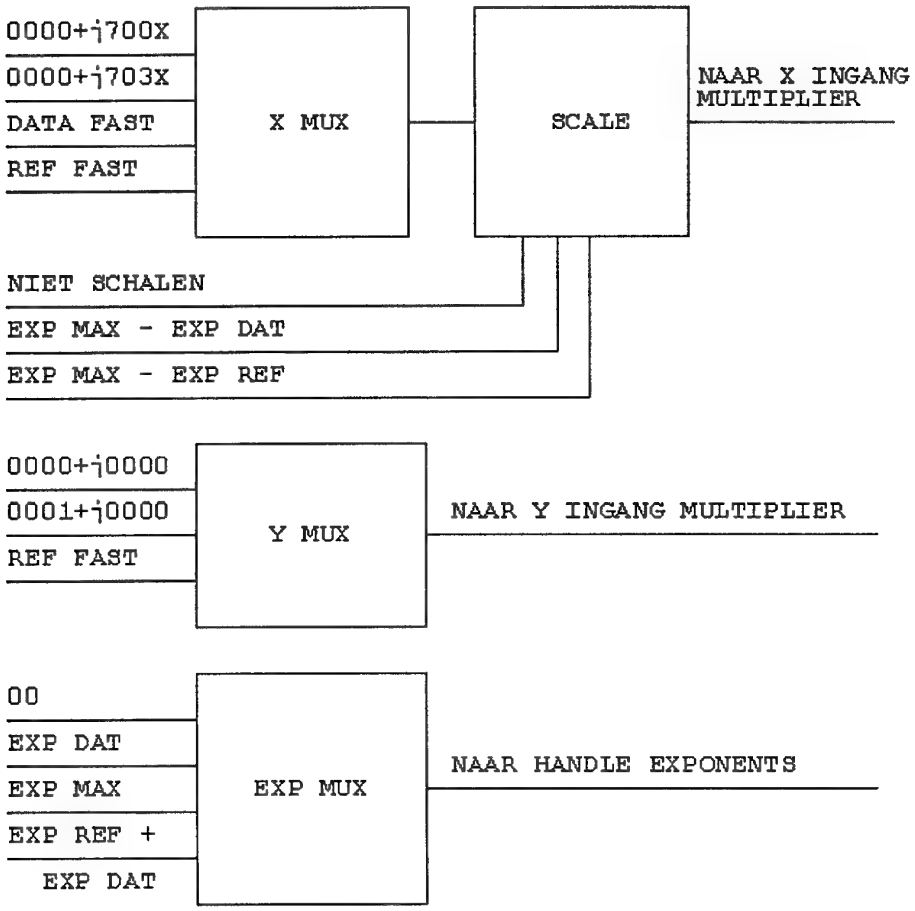
Het bepalen van de meest optimale exponent gaat als volgt. De data staat opgeslagen in het geheugen als een matrix van 256 bij 256 samples. Stel dat de data voorzien is van een exponent die voor een hele rij constant is maar dat de data als kolommen moet worden uitgelezen. De uitgelezen data bevat dan een exponent die per datawoord kan veranderen maar zal moeten worden omgezet naar data met een constante exponent voordat het bij de FFT kan worden aangeboden. Om de dynamiek maximaal te houden moet het grootste getal in een kolom worden opgezocht. De exponent van dit getal moet zo gekozen worden dat de mantisse alle beschikba-

re bits gebruikt. Dit is dan gelijk de meest optimale exponent voor de gehele kolom omdat de dynamiek op deze manier optimaal gebruikt wordt. De optimale exponent kan worden vastgesteld op het moment dat de data in het veld wordt geschreven. Op deze manier is de optimale exponent al beschikbaar zodra de data uitgelezen wordt.

Binnen het blok 'schaling' kunnen vier bewerkingsblokken onderscheiden worden:

- X MUX: Hier wordt bepaald wat via de schaling naar de x ingang van de multiplier gaat.
- SCALE: De data afkomstig van de x multiplexer wordt hier geschaald. Daarna wordt het doorgegeven naar de x ingang van de multiplier.
- Y MUX: Hier wordt bepaald wat naar de y ingang van de multiplier gaat.
- EXP MUX: Hier wordt de nieuwe exponent bepaald. Deze is afhankelijk van data exponent, referentie exponent en de maximum exponent.

De blokken zijn weergegeven in Figuur 4.2.



Figuur 4.2: Blokschematische opbouw 'schaling'.

Deze vier bewerkingsblokken worden bestuurd vanuit state-machine *fcm_state* en het ingangssignaal ZERO. Afhankelijk hiervan worden door de bewerkingsblokken de volgende acties ondernomen, of anders gezegd, *fcm_state* komt in de volgende toestanden:

idle
De schaling staat nog te wachten totdat er wat gaat gebeuren, de instellingen zijn eigenlijk nog niet relevant maar zijn alvast gekozen zoals ze in de volgende state moeten zijn.

define FFT forward
De FFT-chip moet geprogrammeerd worden voordat er data kan worden aangeboden. Om het programmeerwoord aan te bieden bij de FFT-chip wordt bij de x ingang van de multiplier het programmeerwoord van de FFT-chip aangeboden en bij de y ingang van de multiplier het getal één. De multiplier zal het programmeerwoord vermenigvuldigen met één. Hierdoor lijkt de multiplier transparant voor het programmeerwoord. Het programmeerwoord voor de FFT-chip is $(0000_{HEX})+j(7002_{HEX})$ in 3 shifts mode en $(0000_{HEX})+j(700A_{HEX})$ in 2 shifts mode. De exponent uitgang is niet van belang maar wordt op 00_{HEX} gehouden. De shifter mag niet schuiven anders wordt het programmeerwoord verminkt.

column FFT data/read FIFO
De data vanaf de data FIFO (DATA_FAST) moet onveranderd bij de FFT-chip aankomen. Om dit te bereiken zijn de instellingen als volgt:

X MUX: DATA_FAST wordt doorgegeven naar de schaling.
SCALE: de data wordt ongeschaald doorgegeven.
Y MUX: forceert uitgang naar $(0001_{HEX})+j(0000_{HEX})$, de multiplier wordt hierdoor weer transparant.
EXP MUX: de exponent van DATA_FAST wordt doorgegeven.

Bepaalde delen van een datareeks moeten aangevuld worden met nullen. Dit wordt aangegeven met het ZERO signaal, afkomstig van de geheugenbesturing. Er wordt dan overgesprongen naar toestand *zero1*.

zero1
Indien een data reeks afgelopen is, moeten er nullen aan de reeks worden toegevoegd. Om dit te bereiken wordt op de y ingang van de multiplier een nul aangeboden. De instellingen zijn als volgt:

X MUX: De uitgang is eigenlijk niet van belang en wordt daarom hetzelfde gehouden als bij *column FFT data*.
SCALE: Ook hier is de uitgang eigenlijk niet van belang en wordt daarom hetzelfde gehouden als bij *column FFT data*.
Y MUX: Forceert uitgang naar $(0000_{HEX})+j(0000_{HEX})$, waardoor de uitgang van de multiplier ook nul wordt.
EXP MUX: Er is nu niet bekend welke exponent binnen komt op de EXP_DAT_FAST ingang. Deze onbekende exponent wordt wel

doorgegeven naar het volgende blok maar mag daar niet overgenomen worden.

De toestand `zero1` zal beëindigd worden zodra het zero signaal afkomstig van de geheugenbesturing weer afvalt. De toestand *column FFT data/read FIFO* zal beëindigd worden zodra *fcm_state* overgaat naar *row FFT data.*

row FFT data

De data vanaf DATA_FAST moet geschaald worden naar de maximale exponent die bij kolom FFT van de data is voorgekomen. Deze maximale exponent wordt aangeboden op de EXP_MAX ingang. Om dit te bereiken zijn de instellingen als volgt:

X MUX: DATA_FAST wordt doorgegeven naar de scale.

SCALE: De data wordt over (EXP_MAX-EXP_DAT) plaatsen verschoven.

Y MUX: Forceert uitgang naar $(0001_{HEX})+j(0000_{HEX})$ om de multiplier transparant te maken.

EXP MUX: Op de uitgang staat de hoogste exponent die bij de kolom FFT van de data is voorgekomen. Dit is de exponent die binnen komt op EXP_MAX.

Na de toetstand *row FFT data* gaat *fcm_state* over naar de toestand *colum FFT data.*

column FFT referentie

Deze toestand is vergelijkbaar met *column FFT data* met dien verstande dat:

- De referentie naar de x ingang van de multiplier gaat i.p.v. de data.
- Er geen exponent wordt aangeleverd en de POST_SCALE daarom nul wordt gehouden.

zero2

Deze toestand is hetzelfde als *zero1* met dien verstande dat:

- De referentie naar de x ingang van de multiplier gaat i.p.v. de data.
- Er geen exponent wordt aangeleverd en de POST_SCALE daarom nul wordt gehouden.

row FFT referentie

Deze toestand is vergelijkbaar met *row FFT data.* Tijdens *row FFT referentie* wordt de referentie geschaald naar het maximum dat tijdens kolom FFT referentie voorkwam. De instellingen zijn als volgt:

X MUX: REF_FAST wordt doorgegeven naar de scale.

SCALE: De referentie wordt over (EXP_MAX-EXP_REF) plaatsen verschoven.

Y MUX: Forceert uitgang naar $(0001_{HEX})+j(0000_{HEX})$ om de multiplier transparant te maken.

EXP MUX: Op de uitgang staat de hoogste exponent die bij de kolom FFT van de referentie is voorgekomen. Dit is de exponent die binnen komt op EXP_MAX.

Na de toestand *row FFT referentie* gaar de *fcm_state* over naar de toestand *define IFFT.*

define IFFT
Voordat met vermenigvuldigen en rij IFFT verder kan worden gegaan moet de FFT-chip gedefinieerd worden. Hiervoor worden zes klokcycli gereserveerd. De instellingen zijn:

- X MUX: Deze genereert het programmeerwoord voor de FFT chip. Dit is $(0000_{HEX})+j(7032_{HEX})$ in 3 shifts mode en $(0000_{HEX})+j(703A_{HEX})$ in 2 shifts mode.
- SCALE: De data wordt ongeschaald doorgegeven.
- Y MUX: Forceert uitgang naar $(0001_{HEX})+j(0000_{HEX})$ om de multiplier transparant te maken.
- EXP MUX: Uitgang niet van belang maar wordt op 00_{HEX} gehouden.

multiply and row IFFT
De data en de referentie worden met elkaar vermenigvuldigd en daarna bij de IFFT aangeboden. De bijbehorende exponent wordt berekend door de exponent van data en referentie bij elkaar op te tellen (dit is de exponent die bij de data uit de multiplier hoort). Om dit te bereiken zijn de instellingen als volgt:

- X MUX: DATA_FAST wordt doorgegeven naar scale.
- SCALE: De data wordt ongeschaald doorgegeven.
- Y MUX: REF_FAST wordt doorgegeven naar de multiplier.
- EXP MUX: Op de uitgang staat de som van de exponenten van de data en de referentie. Dit is de exponent die hoort bij de data die uit de multiplier komt.

Na de toestand *multiply and row IFFT* gaat *fcm_state* over naar de toestand *column FFT*

column IFFT
De multiplier wordt transparant gemaakt en er wordt een IFFT bewerking gedaan op de resultaten van de voorgaande bewerking. De binnenkomende data wordt weer geschaald naar het maximum van de voorgaande bewerking. De instellingen zijn:

- X MUX: DATA_FAST wordt doorgegeven aan de scale.
- SCALE: De data wordt geschaald naar de waarde die op de EXP_MAX staat.
- Y MUX: Forceert uitgang naar $(0001_{HEX})+j(0000_{HEX})$ om de multiplier transparant te maken.
- EXP MUX: Op de uitgang staat de hoogste exponent die bij de row IFFT van de data is voorgekomen. Dit is de exponent die binnen komt op EXP_MAX.

De instellingen zijn in tabelvorm weergegeven in Tabel 4.1. Hierbij gelden de volgende afkortingen:

id	*idle*
df	*define FFT forward*
cd	*colom FFT dat*
z1	*zero1*
rd	*row FFT dat*
cr	*column FFT ref*
z2	*zero2*
rr	*row FFT ref*
di	*define IFFT*
mi	*multiply and row IFFT*
ci	*column IFFT*

Tabel 4.1: Instellingen in tabelvorm.

X MUX	*id*	*df*	*cd*	*z1*	*rd*	*cr*	*z2*	*rr*	*di*	*mi*	*ci*
0000+j700x		x									
DATA_FAST			x	x	x					x	x
REF_FAST						x	x	x			
0000+j703x	x								x		
Y MUX	***id***	***df***	***cd***	***z1***	***rd***	***cr***	***z2***	***rr***	***di***	***mi***	***ci***
0001+j0000	x	x	x		x	x		x	x		x
0000+j0000				x			x				
REF_FAST										x	
EXP MUX	***id***	***df***	***cd***	***z1***	***rd***	***cr***	***z2***	***rr***	***di***	***mi***	***ci***
00_H	x	x				x	x		x		
EXP DATA_FAST			x	x							
EXP MAX					x			x			x
REF+DAT										x	
SCALE	***id***	***df***	***cd***	***z1***	***rd***	***cr***	***z2***	***rr***	***di***	***mi***	***ci***
NIET SCHALEN	x	x	x	x		x	x		x	x	
EXP_MAX-DAT_MAX					x						x
EXP_MAX-EXP_REF								x			

4.7 Vermenigvuldiger

De vermenigvuldiger is een aangekochte component van Plessey. De vermenigvuldiger heeft slechts één besturingssignaal nodig. Dit besturingssignaal houdt verband met de datagroei die in de vermenigvuldiger optreedt. Als er namelijk twee 16 bits complexe getallen met elkaar vermenigvuldigd worden dan is het resultaat een 32 bits complex getal. Omdat er in de FCM met 16 bits complexe getallen gewerkt wordt, worden alleen de bovenste 16 bits van het resultaat gebruikt. Indien de vermenigvuldiger echter als een transparant doorgeefluik (voor bijvoorbeeld het definitiewoord van de FFT-chips) wordt gebruikt dan zijn alleen de onderste 16 bits van het resultaat van belang. Met het signaal MPY_MODE worden de bovenste 16 bits van het resultaat geselecteerd als *fcm_state* zich in de toestand *multiply and row FFT* bevindt terwijl in alle andere gevallen de onderste 16 bits geselecteerd worden.

4.8 FFT

Om de FFT bewerking uit te voeren worden twee chips van Plessey parallel gezet. Ze moeten door een beperkt aantal signalen bestuurd worden. Voordat de FFT aan de dataverwerking begint, moet hij gedefinieerd worden. Het blok 'schaling' wekt hiertoe een definitiewoord op. Deze komt via de vermenigvuldiger bij de FFT-chips terecht. Zodra dit woord enige klokcycli stabiel is wordt het N_DEF signaal geactiveerd om het definitiewoord in te laten nemen door de FFT-chips.

De datareeksen die bij de FFT-chips arriveren worden afwisselend door de ene FFT-chip en de andere FFT-chip ingenomen. Dit wordt gerealiseerd door alternerend bij de ene FFT-chip of bij de andere FFT-chip het N_INEN signaal te activeren. Om te voorkomen dat in een foutsituatie beide FFT-chips tegelijk data op de uitgang willen zetten (waardoor de uitgang van een FFT-chip zou kunnen beschadigen) is er een controle ingebouwd. Indien beide FFT-chips data op de bus willen zetten, wordt één FFT-chip afgeschakeld.

4.9 Exponent

De databewerkingen multiply, FFT en IFFT zorgen voor een groei van de data en dus een verandering van de exponent. Het blok 'handle exponent' bepaalt de nieuwe exponent na deze databewerkingen.

Uit het blok 'schaling' komt data met een bijbehorende exponent tevoorschijn. De exponent gaat direct naar 'handle exponent' en de data komt via de multiplier en de FFT-chip bij 'handle exponent' aan. De data loopt vertraging op door het langere pad en daarom moet ook de exponent vertraagd worden. In het blok 'FIFO' wordt de exponent afkomstig van de schaling vertraagd. De binnenkomende expo-

nent wordt in een FIFO van acht bits breed en drie woorden diep opgeslagen. Een exponent wordt in de FIFO opgeslagen als er een exponent valid (EXP_S_VAL) binnen komt en er wordt een volgende exponent uit het FIFO gehaald zodra er een nieuwe datareeks uit de FFT-chip komt (N_DAV actief).

Door verschillende oorzaken moet de exponent gecorrigeerd worden met een constante factor. De constante factor is afhankelijk van de volgende punten:

- hoe is de vermenigvuldiger ingesteld.
- werkt de FFT-chip in 2 of 3 shifts mode.
- wordt er een FFT of een IFFT bewerking gedaan.
- wordt er overlap/add gedaan.

De correctiefactor voor een FFT bewerking in 2 shifts mode is 8, de correctiefactor voor een FFT bewerking in 3 shifts mode is 11. Voor de IFFT bewerking geldt hetzelfde. In de mode overlap/add kan nog een groei van één bit ontstaan. In deze fase kan de exponent alvast met één verhoogd worden om overflow tijdens overlap/add te voorkomen. Het resultaat van (correctie factor + exponent van de scale - FFT shift) is de exponent die samen met de data uit de FFT geschreven wordt in het geheugen.

De mantisse van de getallen die naar het geheugen geschreven worden kunnen voor een aanzienlijk deel bestaan uit tekenbits. Om deze reden wordt een meest optimale exponent bepaald waarbij er in een rij of een kolom minimaal één mantisse is met slechts één tekenbit. Om deze optimale exponent te bepalen wordt voor zowel de reële als het imaginaire woord uit de FFT-chip het aantal teenbits bepaald. De kleinste van de twee waarden wordt gebruikt om de optimale exponent te bepalen. Deze wordt vervolgens vergeleken met het resultaat van dezelfde bewerking voor de voorgaande reeksen. Deze staat opgeslagen in een FIFO (max FIFO). De grootste van de twee wordt weer teruggeschreven in de max FIFO.

4.10 Rechterkant van het geheugen

De data die bij de FFT gegenereerd wordt, komt via het blok overlap/add aan bij de rechterkant van het geheugen. Een grote state machine (*right*) bepaalt hoe de data in het geheugen geschreven zal worden. Afhankelijk van de toestand van de state-machine *right* worden adres counters aangestuurd en wordt het geheugen voorzien van besturingssignalen.

Alleen tijdens de mode overlap/add wordt geen gebruik gemaakt van de state-machine *right* maar van de state-machine *right_out*. Zodra aan de rechterkant van het geheugen een compleet veld van 256 bij 256 datawoorden is weggeschreven kan de state-machine *fcm_state* overgaan naar de volgende state. Dit wordt aangegeven door het signaal STATE_RDY te activeren.

4.11 Overlap/add

De functionaliteit van het blok overlap/add wordt verwezenlijkt door twee FPGA's. Eén FPGA verwerkt de reële data en één FPGA verwerkt de imaginaire data. Indien de schakeling zich in de toestand *overlap/add* bevindt dan worden de datastroom uit het geheugen en de datastroom uit de FFT-chip in dit blok bij elkaar opgeteld. Om de mantissen bij elkaar op te kunnen tellen worden ze eerst geschaald naar dezelfde exponent. Het resultaat wordt aangeboden bij de barrel shifter.

4.12 Barrel shifter

De data die door de overlap/add gegenereerd wordt heeft een variërende exponent. De data die door de FCM gegenereerd moet worden zal echter over een constante exponent moeten beschikken. De data wordt daarom in het blok 'barrel shifter' geschaald naar een exponent die middels een programmeerwoord ingesteld kan worden. Het gevaar bestaat echter dat de exponent in het programmeerwoord te klein wordt gekozen. In die gevallen wordt de mantisse omhoog geschaald totdat het reële of het imaginaire deel nog één sign bit heeft. Hierdoor is de mantisse te klein maar heeft nog de goede fase. Dit lijkt voor toepassing in een SAR systeem het beste compromis. Het blok 'barrel shifter' is geïmplementeerd door eerst het aantal sign bits in de reële en het imaginaire deel te bepalen. De kleinste van de twee is de maximale verschuiving die op de mantisse mag worden toegelaten. Ook wordt het verschil tussen de exponent in het programmeerwoord en de exponent afkomstig van de 'overlap/add' bepaald. Dit verschil (met een maximum zoals eerder bepaald) is de verschuiving die op de mantissen wordt toegepast. Het resultaat wordt in de uitgangs-FIFO geschreven.

4.13 UitgangsFIFO

Als interface naar de buitenwereld beschikt de FCM over een FIFO. Hierdoor kan er zonder problemen overgegaan worden van de 20MHz klok naar de dataklok, die elke mogelijke frequentie kan hebben. Het moment waarop de data in de FIFO wordt geschreven, wordt door dezelfde logica bepaald die ook bepaalt wanneer er data uit de rechterkant van het geheugen gelezen wordt. Tussen het lezen uit de rechterkant van het geheugen en het schrijven in de uitgangs-FIFO zit namelijk alleen een vaste pipeline delay. De FCM zal data op de uitgang gaan zetten zodra alle samples in de uitgangs-FIFO staan en zodra de FCM aan de beurt is om data op de bus te mogen zetten (dit gebeurt volgens het mechanisme zoals in de paragraaf 4.2 beschreven is). De lengte van de datareeksen wordt bepaald door middel van programmeerwoorden. Omdat de FCM echter ondertussen al begonnen is met het verwerken van nieuwe data, zijn de programmeerwoorden al ververst. De

besturing van de uitgangs-FIFO zal dus gebruik moeten maken van de oude programmeerwoorden.

5. Testresultaten

Om de werking van de FCM te controleren zijn een aantal tests op de schakeling uitgevoerd. Hier volgt een opsomming van de 'onvolkomenheden' die gevonden zijn tijdens het testen van de FCM. Voorts zijn er nog een groot aantal tests uitgevoerd die goede resultaten hebben opgeleverd. Deze staan hier echter niet vermeld.

- *Gebruiksklaar maken van de omgeving*
  Om de FCM op te kunnen nemen in de Real-Time SAR omgeving (zie [3]) is er een aanpassing gedaan op de printkaart die de data aanlevert voor de FCM (de zgn. puls compressor interface). Op de FCM is het signaal ENIN geinverteerd i.v.m. de aanwezige kastbedrading.
- *Matrix omzetten van 32x32 naar 256x256*
  In het datageheugen en het referentiegeheugen staan de getallen opgeslagen als een 256×256 matrix. Voor de simulaties is deze matrix echter aan de grote kant, het zou resulteren in lange simulatietijden. Om deze reden is er tijdens de simulaties gebruik gemaakt van een 32×32 matrix. Nu de schakeling in bedrijf wordt gesteld moeten de Altera FPGA's omgezet worden naar een 256×256 matrix. Ondanks dat hiervoor geen extra logica nodig is, is het toch een moeizaam proces geweest. De problemen lagen bij het 'fitten' van de nieuwe logica in hetzelfde device en met dezelfde aansluitingen naar de printkaart.
- *Thermische problemen*
  In deze schakeling worden de FFT-chips van Plessey op 40 MHz bedreven. In de testopstelling (zonder koellichaam en zonder geforceerde koeling) blijken de FFT-chips te heet te worden. Om deze reden is de kloksnelheid van de schakeling gehalveerd.
- *Ontbreken van de referentie bus*
  In het Cadnetics PCB ontwerpsysteem is het mogelijk om meerdere signalen samen te nemen in een bus (via de zgn 'conts' file). Bij het aansluiten van de referentie signalen op een Altera FPGA is de naam van de bus gebruikt i.p.v. de naam van de signalen. Het resultaat is dat de referentie signalen niet aangesloten zaten. De controles die binnen het Cadnetics systeem bestaan ('verify', 'dance' en 'drink') geven geen waarschuwing als dit voorkomt. Richting de leverancier van Cadnetics is een verzoek uitgegaan om dit probleem te verhelpen.
- *Connector naar het referentie geheugen*
  De FCM bestaat uit twee printkaarten, de basisprintkaart en een opsteek printkaart t.b.v. het referentiegeheugen. Tussen deze twee printkaarten zitten 9 connectoren om 144 signalen over te dragen. De printkaarten zijn als twee aparte projecten behandeld en zodoende bestaat er ook geen enkele mogelijkheid om automatisch te laten controleren of de signalen tussen de twee printkaarten goed worden overgedragen. Tijdens het routen van de printkaart zijn de connectoren fout geplaatst, dit is verder niet meer opgemerkt en heeft geresulteerd in 96 aparte draadjes tussen de twee printkaarten.

- *Gebreken van de testdop*
  Een PGA pennetje van de testvoet was afgebroken en achtergebleven in een andere voet. Ean ander PGA pennetje van de testvoet was niet doorverbonden op de testvoet.
- *Verkeert uitlezen exponent FIFO*
  Er kunnen meerdere data reeksen worden aangeboden bij de FCM. Elke reeks gaat vergezeld van een eigen exponent. De data wordt eerst in een FIFO opgeslagen. Bij het uitlezen van de FIFO werd niet per datareeks een exponent uitgelezen maar er werd per kolom een nieuwe exponent uitgelezen. Dit is verholpen.

Alle onvolkomenheden zijn verholpen middels het leggen van een draadje of het opnieuw programmeren van een Altera FPGA.

6. Evaluaties en Conclusies

In het project Fast Convolution Module is vooral geprobeerd om door middel van gestructureerde ontwerpmethodieken risico's in het ontwerpproces te minimaliseren. Deze gestructureerde ontwerpmethodieken worden in de toekomst steeds essentiëler omdat de complexiteit van DSP hardware (waaronder die van de FCM) steeds groter wordt. Belangrijk is hierbij dat in een vroeg stadium van het ontwerp fouten worden gedetecteerd, bijvoorbeeld door de ontwerper te voorzien van terugkoppeling uit reviews met projectleider en mede-ontwerpers. Om deze reviews zo effectief mogelijk te benutten is het belangrijk om op elk ontwerpniveau een heldere en eenduidige manier van ontwerpen in te stellen.

In het geval van de Fast Convolution Module is voor het functionele ontwerp gekozen om gebruik te maken van de Structured Analysis Method. Deze methode legt in feite het denkproces van de ontwerper eenduidig en hiërarchisch vast. Hij 'dwingt' een ontwerper om het hele ontwerp te verifiëren als ergens een kleine verandering in het ontwerp heeft plaatsgevonden. Uit ervaring blijkt dat deze 'kleine' veranderingen in de praktijk grote gevolgen kunnen hebben in het al dan niet functioneren van het uiteindelijke ontwerp.

Voor het detail ontwerp is gebruik gemaakt van de CAE tools die op TNO-FEL aanwezig zijn, namelijk de Dazix tool (Printed Circuit Board ontwerp, simulatie van de schakeling) en de Altera ontwerp tool (programmeerbare logica). De hele control van de schakeling en een gedeelte van de datapath (o.a. schaling en overlap/add) zijn in Altera componenten geïmplementeerd en gesimuleerd. Deze tool biedt de hardware ontwerper de gelegenheid om zijn ontwerp zowel op functioneel als op timing gedrag uitgebreid te simuleren. In principe is er geen limiet aan het aantal Altera componenten dat als een geheel gesimuleerd kan worden.

Vervolgens zijn de Altera componenten tezamen met de Commercial Off-The-Shelf (COTS) componenten (FFT chips, multiplyer, geheugen componenten) ingevoerd in de Dazix tool. In principe zouden we nu de gehele schakeling hebben kunnen simuleren (zgn. board-level simulatie). De grote hoeveelheid Altera componenten binnen de schakeling (9) leverde echter problemen op voor Dazix. Volgens de leverancierspecificatie is er geen limiet aan het aantal Altera componenten binnen een te simuleren schakeling, in de praktijk blijkt dat met meer dan twee Altera componenten het systeem crasht. Helaas is er voor dit probleem geen bevredigende oplossing gevonden. Om toch zoveel mogelijk gesimuleerd te krijgen, is het ontwerp in kleinere delen opgesplitst. In de toekomst zal met dit probleem rekening gehouden moeten worden, door bijvoorbeeld het ontwerp van meet af aan al in 'simuleerbare' delen op te splitsen. Een beter alternatief is om het gehele project met behulp van de hardware beschrijvingstaal VHDL te ontwerpen, en vervolgens een simulatie tool te gebruiken die de board-level simulatie wel aan

kan. De VHDL code kan vervolgens gecompileerd worden ten behoeve van Altera implementatie.

Een bijkomend voordeel van een VHDL beschrijving is dat relatief snel een vervolgtraject kan worden ingezet ten behoeve van de miniaturisatie. In het concept van de Fast Convolution Module is impliciet rekening gehouden met het feit dat in de toekomst miniaturisatie mogelijk moet zijn. Door een tweejarige AIO (TWAIO) van de Universiteit Twente is recentelijk aangetoond dat met de huidige Aplication Specific Integrated Circuit (ASIC)-technologie (0.8 μ CMOS) de volledige functionaliteit van de FCM met behoud van de performance in één chip kan worden geïmplementeerd [2].

Momenteel wordt door TNO-FEL gewerkt aan een beschrijving van deze chip in VHDL. Hiermee is een belangrijke stap gezet in de realisatie van een demonstrator Real-Time SAR Processor. Met deze demonstrator toont TNO-FEL aan dat real-time SAR processing bedreven kan worden zonder verlies aan performance of functionaliteit, met een hardware processor die aanzienlijk kleiner is en minder vermogen verbruikt dan de huidige hardware processoren.

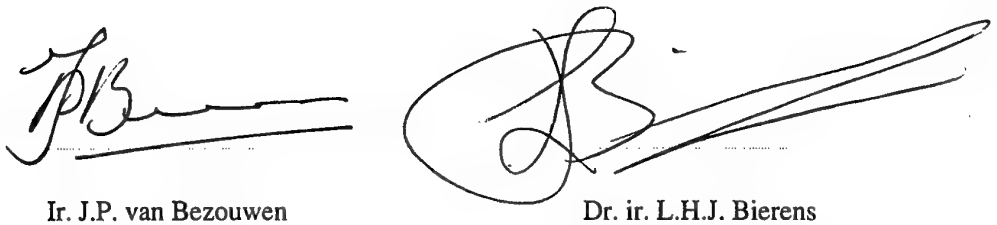
7. Referenties

[1] TNO-rapport FEL-93-B113
A Real-Time Convolution Algorithm and Architecture with Applications in SAR Processing
L.H.J. Bierens
oktober 1993

[2] University of Twente / TNO-FEL
Design of a Convolution Chip for Real Time SAR processing
H.T.J. Zwartenkot
november 1996

[3] TNO-rapport FEL-94-A292
Testbedomgeving real time SAR
Ir. A.W.P. van Heijningen, Ing. B. Dondertman, Ing. R. v.d. Bos, Ing. C. van 't Wout, Ing. R. Prevo, C.D. Pieterse, A.A. Romein
mei 1995

8. Ondertekening



Ir. J.P. van Bezouwen
Groepsleider

Dr. ir. L.H.J. Bierens
Projectleider/Auteur

Bijlage A Keuze Componenten

In Tabel A.1 zijn de componenten weergegeven die zijn gebruikt voor de implementatie van de FCM, met daaronder een korte motivering van een aantal keuzes.

Tabel A.1: Componentkeuze.

Functionaliteit	Beschrijving	Type	Aantal
Data FIFO	32Kx9 syncFIFO	IDT 72271	4
Referentie FIFO	32Kx9 syncFIFO	IDT 72271	4
Exponent FIFO	256x8 syncFIFO	IDT72200L35TP	1
Data geheugen	32Kx8 DPRAM	IDT 7007 S 25 PF	10
Referentie geheugen	32Kx8 SRAM	CY7C199-20VC	10
Schakel buffers	FPGA	EPM7192EGC-12	2
Scale	FPGA	EPM7192EGC-12	2
Multiplier	complex multiplier	PDSP16116A BOAC	1
FFT-chip	FFT processor	PDSP16510A COAC	2
Maximum	FPGA	EPM7192EGC-12	1
Exponent FIFO	256x8 syncFIFO	IDT72200L35TP	1
Overlap/add	FPGA	EPM7192EGC-12	1
Output FIFO	32Kx9 syncFIFO	IDT 72271	6
Uitgangsbuffer	octal buffer	74F245D	7
Klokbuffer	octal buffer	74F245D	1

Enige motivering en/of toelichting:

- *Data, referentie en output FIFO:*
  Om de interface naar de buitenwereld eenvoudig te houden zullen de FIFO's synchroon moeten zijn. Indien de FIFO's op de print moeten worden gemonteerd moet de capaciteit minimaal 32k x 8 of 16k x 16 zijn. Indien de FIFO's een kleinere capaciteit hebben moet een oplossing worden gekozen met modules, om zo de afmetingen te beperken. De enige mogelijkheid om de componenten op de print te kunnen monteren bieden de IDT72271 32k x 9 syncFIFO en de IDT72265LB 16k x 18 syncFIFO. Er wordt echter verwacht dat deze componenten voor het eerst geproduceerd worden tussen augustus en oktober. Voorkeur gaat uit naar de 32k x 9 bits versie omdat de ingang dan slechts tegen één IC hoeft te praten ipv twee IC's bij de 16k x 18 bits oplossing. De IDT72271 wordt leverbaar in een 64 pins TQFP behuizing (afmetingen: 16 x 16mm). Simulatie modellen zijn nog niet aanwezig.
- *Exponent FIFO:*
  De IDT72200 is een FIFO van 256 woorden diep en 8 bits breed. Om aan een klokfrequentie van 20MHz te voldoen is gekozen voor de 35ns versie. Indien in de toekomst de behoefte bestaat om de data in te nemen op een hogere frequentie dan kan er voor een snellere FIFO gekozen worden, er zijn ook IDT72200 componenten leverbaar met een klokcyclustijd van 25, 20, 15 of 12ns. De FIFO is leverbaar in een plastic DIP behuizing (afmetingen: 35 x 8mm, extensie TP) of in een ceramische DIP behuizing(afmetingen: 36 x 8mm, extensie TC). In-

dien de behuizing te groot is kan worden gekozen voor de IDT72201. Het enige verschil met de IDT71200 is dat hij 9 bits breed is en dat hij verkrijgbaar is in een PLCC behuizing(afmetingen: 15 x 12mm, extensie J) of in een LCC behuizing(afmetingen: 11 x 14mm, extensie L). Omdat slechts 2 FIFO's nodig zijn is de ruimtewinst van de kleinste ten opzichte van de grootste minimaal en wordt gekozen voor de IDT72200L35TP, 256 x 8 bit syncFIFO, low power, 35ns klokcyclustijd in een plastic DIP behuizing. Een model is niet aanwezig in de logic modelling bibliotheek. Een component die grote gelijkenis vertoont met de 72200 is de 72205LB. Qua functionaliteit zijn deze IC's hetzelfde, alleen is de ene 8 bits breed en de andere 18.

- *Data, toggle en referentie geheugen:*
  Voor elk geheugen is 64k x 40 bits nodig. Door de 64k op te splitsen in kleinere brokken kan energie bespaart worden. De brokken die niet gebruikt worden kunnen dan in een power down mode gezet worden. Het liefste zou gekozen worden voor synchroon SRAM bijvoorbeeld de CY173. Deze heeft echter een ongelukkig grote behuizing zodat deze afvalt als kandidaat. Als a-synchroon alternatief kan worden gekozen voor de CY7C199.
- *Schakel buffers:*
  Een kompakte oplossing biedt een ALTERA chip met minimaal 106 pennen. De functionaliteit zal geen problemen geven. De keuze is gevallen op een EPM7192EGC-12. De E versie heeft 6 OE lijnen; alle zes zijn nodig, waardoor de gewone versie afvalt als kandidaat. Als behuizing is voor PGA gekozen. De snelheid van ingang naar uitgang voor combinatoriek is 12ns.

Bijlage B Test FFT-chip

Om de bruikbaarheid van de PDSP16510 (FFT chip) te bepalen voor het projekt KLu Fast Convolution Module is een test opgezet. Vragen die hierbij beantwoord moeten worden zijn:

- Kan de FFT-chip drie reeksen tegelijk afwerken.
- Werkt de FFT-chip nog betrouwbaar als de verwerkingsklok wordt opgedraaid tot 37.5MHz.
- Met hoeveel dummy strobes moet rekening worden gehouden tijdens het uitlezen van de chip.
- Wat is de verwerkingstijd.

Hierbij is gebruikt gemaakt van de volgende testomgeving:

- De FFT-chip bevindt zich op een testprint.
- De testsignalen, waaronder ook de inname- en afgifteklok worden opgewekt door de HP 16500B.
- Voor de systeemklok wordt gezorgd door de PM 5786B.
- Het uitgangssignaal wordt zichtbaar gemaakt m.b.v. de D/A converter van Jan Voltman en een oscilloscoop.

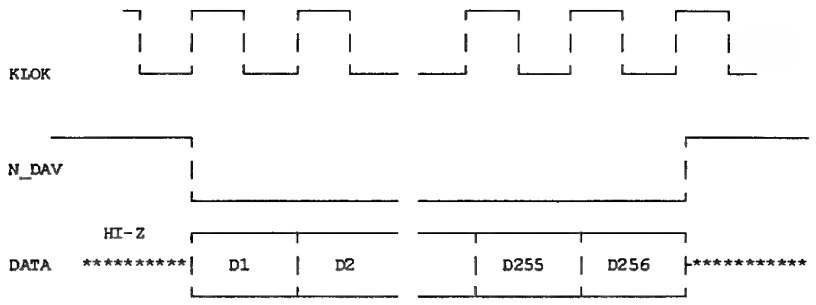
Test1: Meerdere reeksen in de FFT-chip en op een door de gebruiker bepaald moment uitlezen.

Definitie woord: $700A_{HEX}$
Inname klok: 20MHz
Verwerkingsklok: 37MHz
Afgifte klok: 20MHz
Testvektor: Zie benden.
Resultaat: Het meerdere keren uitlezen van de FFT-chip op een door de gebruiker bepaald moment lukt niet omdat de FFT-chip niet meer weet voor welke reeks de NOT_DEN bestemd is.

Test2: Meerdere reeksen in de FFT-chip met uitlezen op een door de FFT-chip bepaald moment.

Definitie woord: 7002_{HEX} en $700A_{HEX}$
Inname klok: 20MHz
Verwerkingsklok: ± 31MHz tot ± 77MHz
Afgifte klok: 20MHz
Testvektor: Zie beneden.
Resultaat:
- De FFT-chip krijgt nu meerdere datareeksen tegelijk te verwerken en kan deze direkt afgeven zodra de verwerking klaar is (NOT DEN is kontinue aktief).
- Zowel 2- als 3-shifts mode zijn beproefd.

- Indien er slechts een klokcyclus pauze tussen definitie en start inname zit zijn er ook geen problemen.
- Tussen definitie en start afgifte eerste reeks zit ongeveer 35,3µs (niet nauwkeurig te bepalen).
- M.b.v. het NOT_DAV signaal kan bepaald worden wanneer geldige data op de uitgang staat.
- De N_DAV uitgang en de data staan in tijd gezien als volgt:



Testvector test1
Voor definitie en inlezen van twee cycli wordt het volgende patroon 1 maal gegenereerd:

```
PATTERN GENERATOR PROGRAM LISTING
          Instr      N_DEF    N_INEN   N_DEN    DATA15
                     Binary   Binary   Binary   Binary
   2   REPEAT   5     1         "        "        "
   3                  "         0        "        "
   4                  "         "        "        1
   5   REPEAT 254     "         "        "        0
   6                  "         1        "        "
   7                  "         0        "        "
   8                  "         "        "        1
   9   REPEAT 254     "         "        "        0
  10                  1         1        1        0
```

Vervolgens wordt het volgende deel oneindig maal aangeboden bij de FFT chip:

```
PATTERN GENERATOR PROGRAM LISTING
          Instr      N_DEF    N_INEN   N_DEN    DATA15   TRIG
___                  Binary   Binary   Binary   Binary   Hex
   1                  "         0        0        "        0
   2                  "         "        "        1        "
   3   REPEAT 254     "         "        "        0        "
   4   REPEAT   4     "         1        "        "        "
   5                  1         1        1        0        "
```

Testvector test2

PATTERN GENERATOR PROGRAM LISTING

	Instr	N_DEF	N_INEN	N_DEN	DATA15	TRIG
		Binary	Binary	Binary	Binary	Binary
0		1	1	0	0	1
1		0	"	"	0	0
2	REPEAT 5	1	"	"	"	"
3		"	0	"	"	"
4		"	"	"	1	"
5		"	"	"	0	"
6		"	"	"	"	"
7	REPEAT 252	"	"	"	"	"
8	REPEAT 8	"	1	"	"	"
9	REPEAT 256	"	"	"	"	"
10	REPEAT 8	"	"	"	"	"
11		"	0	"	"	"
12		"	"	"	"	"
13		"	"	"	1	"
14		"	"	"	0	"
15	REPEAT 252	"	"	"	"	"
16	REPEAT 8	"	1	"	"	"
17	REPEAT 256	"	"	"	"	"
18	REPEAT 8	"	"	"	"	"
19		"	0	"	"	"
20		"	"	"	"	"
21		"	"	"	"	"
22		"	"	"	1	"
23		"	"	"	0	"
24		"	"	"	"	"
25	REPEAT 250	"	"	"	"	"
26	REPEAT 8	"	1	"	"	"
27	REPEAT 256	"	"	"	"	"
28	REPEAT 8	"	"	"	"	"
29		"	0	"	"	"
30		"	"	"	"	"
31		"	"	"	"	"
32		"	"	"	"	"
33		"	"	"	1	"
34		"	"	"	0	"
35		"	"	"	"	"
36	REPEAT 249	"	"	"	"	"
37	REPEAT 8	"	1	"	"	"
38	REPEAT 256	"	"	"	"	"

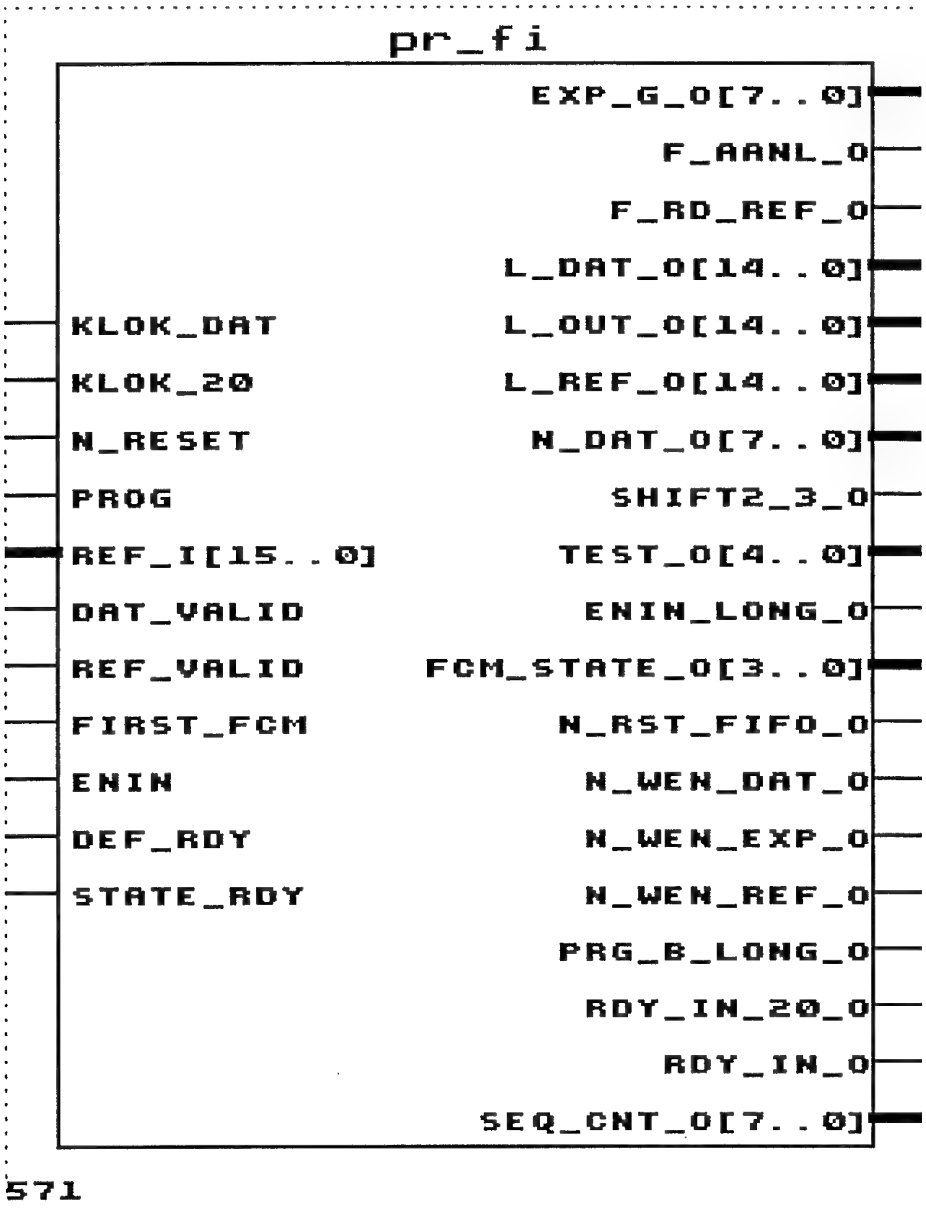
39	REPEAT 8	"	"	"	"	"
40		"	0	"	"	"
41		"	"	"	"	"
42		"	"	"	"	"
43		"	"	"	"	"
44		"	"	"	"	"
45		"	"	"	1	"
46		"	"	"	0	"
47		"	"	"	"	"
48	REPEAT 248	"	"	"	"	"
49	REPEAT 8	"	1	"	"	"
50	REPEAT 256	"	"	"	"	"
51	REPEAT 8	"	"	"	"	"
52		"	0	"	"	"
53		"	"	"	"	"
54		"	"	"	"	"
55		"	"	"	"	"
56		"	"	"	"	"
57		"	"	"	"	"
58		"	"	"	1	"
59		"	"	"	0	"
60	REPEAT 248	"	"	"	"	"
61	REPEAT 8	"	1	"	"	"
62	REPEAT 256	"	"	"	"	"
63	REPEAT 8	"	"	"	"	"

Bijlage C Globale beschrijving van de FPGAs

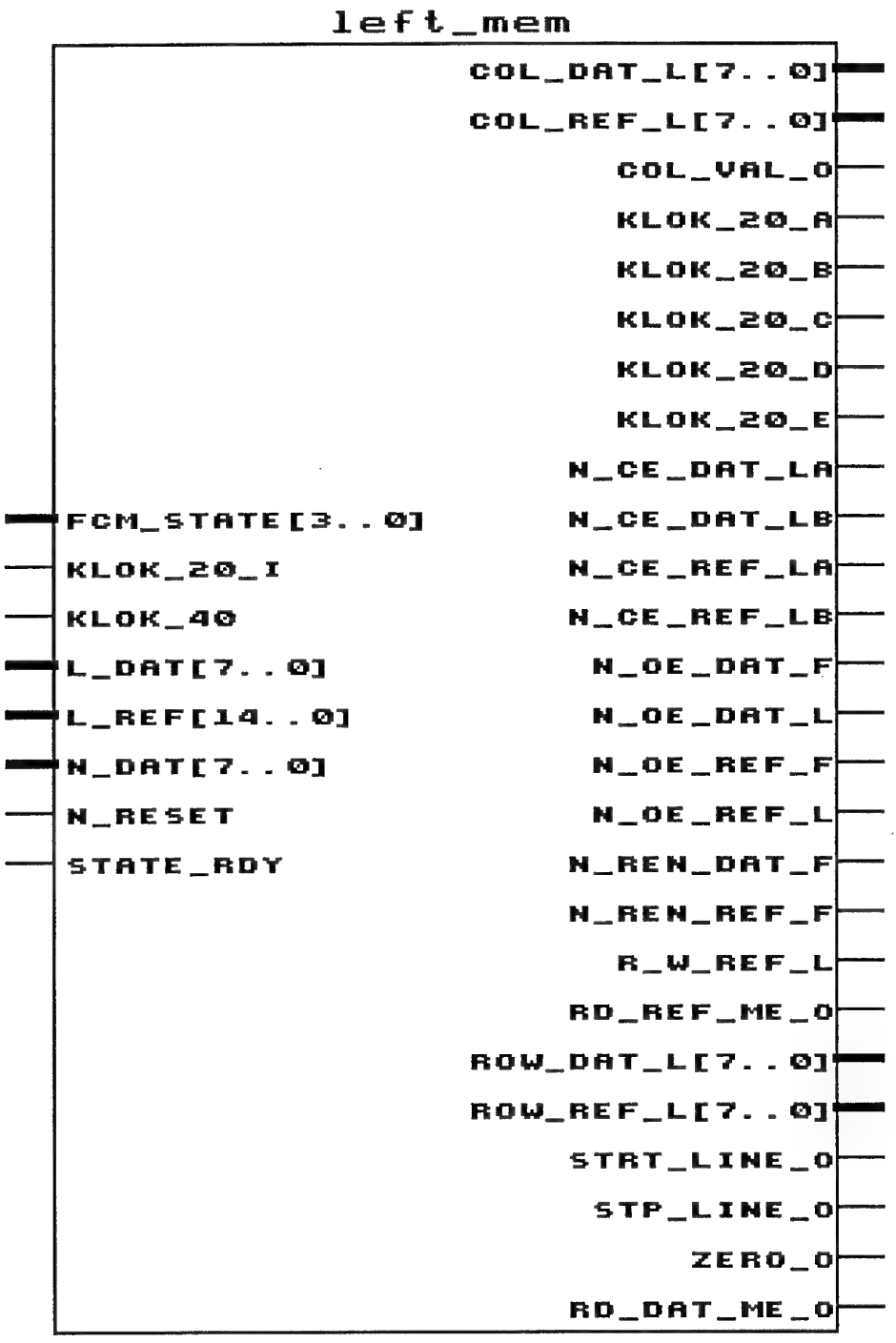
In Figuur C.1 t/m Figuur C.9 zijn de FPGAs weergegeven, met hun belangrijkste in- en uitgaande signalen. Tabel C.1 geeft de signaalnamen zoals in het rapport worden gebruikt, alsmede de namen die in de FPGA source code worden gebruikt.

Tabel C.1: Conversietable van signaalnamen zoals ze in dit rapport worden gebruikt naar signaalnamen gebruikt in desource code van de FPGAs.

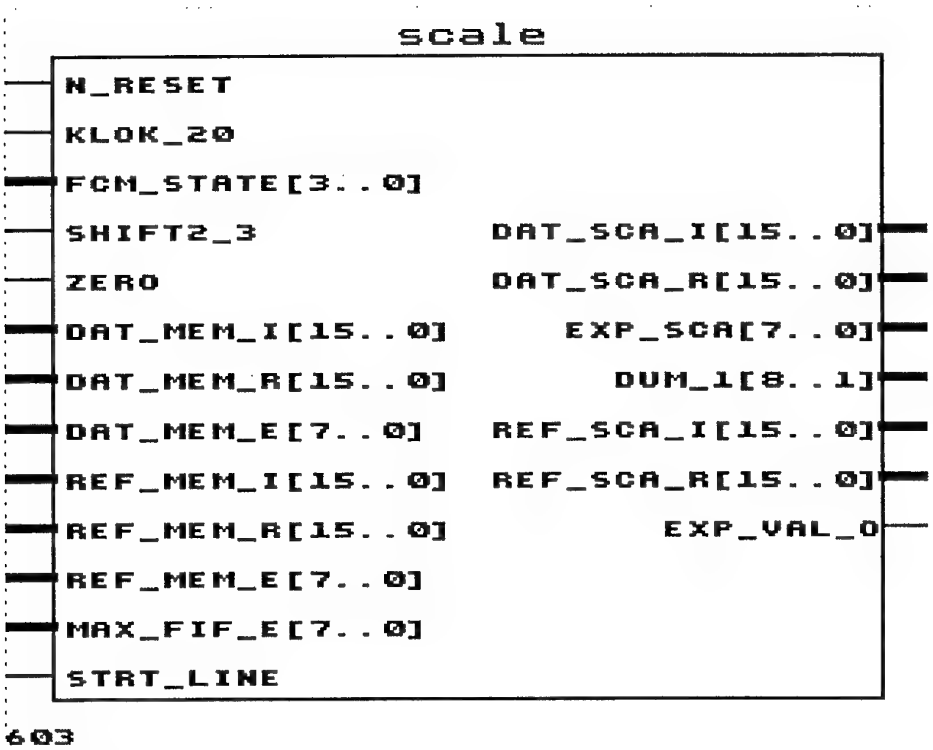
Signaalnamen rapport	Signaalnamen FPGAs	Signaalnamen rapport	Signaalnamen FPGAs
PROG	prog	L_DAT	l_dat
START	-	L_OUT	l_out
LDD	ldx	N_DAT	n_dat
STOPD	stop_x	F_AANLOOP	f_aanl
DVALID	dat_valid	F_REF	f_rd_ref
RVALID	ref_valid	2_3SHIFT	shift2_3
LDR	ldh	DAT_SEC	-
STOPR	stop_h	REF_SEC	-
DAT	x_h, x_i	PROG_BEEN	prog_been
EXPD	exp_in	OVAD_OUT	ovad_uit
REF	h_r, h_i	READ_PROG	read_prog
KLOKIN	klok_dat	ENIN_LONG	enin_long
KLOKOUT	klok_dat	SEQ_CNT	seq_cnt
RESET	n_reset	ZERO	zero
ENIN	enin	DATA_FAST	data_mem
READYIN	rdy_in	REF_FAST	ref_mem
FIRSTFCM	first_fcm	EXP_MAX	max_fif_e
OUT	r_r, r_i	EXP_DAT	dat_mem_e
EXPO	exp_r	POST_SCALE	exp_sca
STARTO	-	MPY_MODE	mpy_mode
LDO	-	N_DEF	n_def
STOPO	-	N_INEN	n_inen
OVALID	n_rvalid	N_DEN	n_den
ENOUT	enout	EXPS_VALID	exp_val
READYOUT	ready_out	STATE_RDY	state_rdy
L_REF	l_ref		



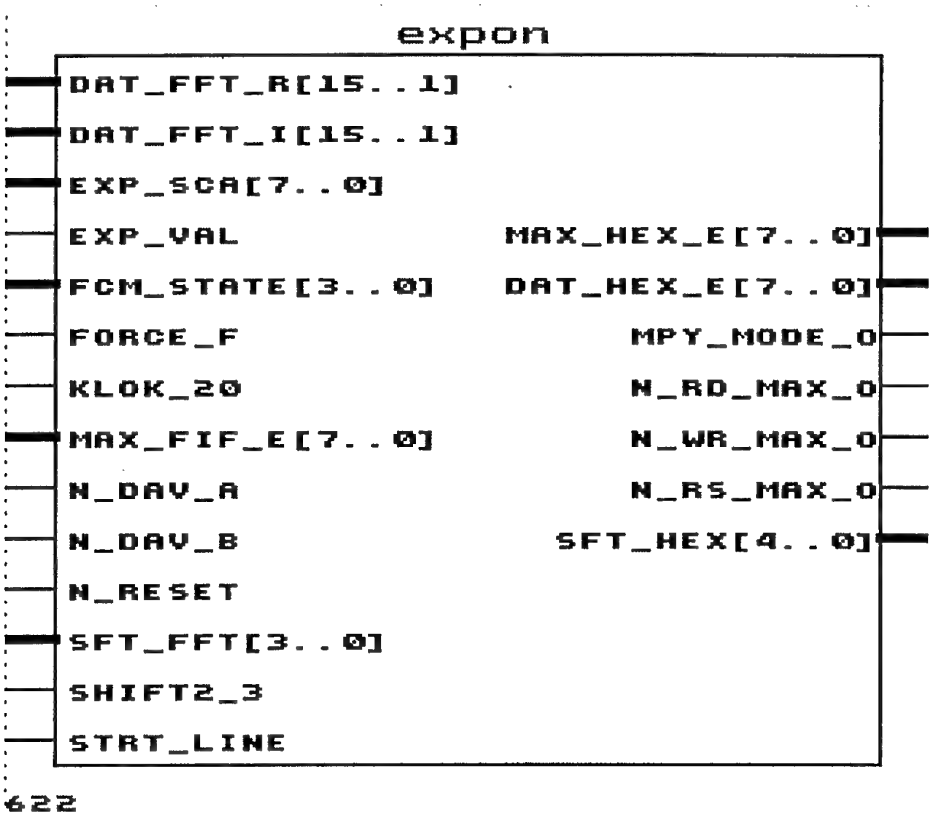
Figuur C.1: De FPGA die de programmeerwoorden inneemt en de ingangsfifo bestuurt.



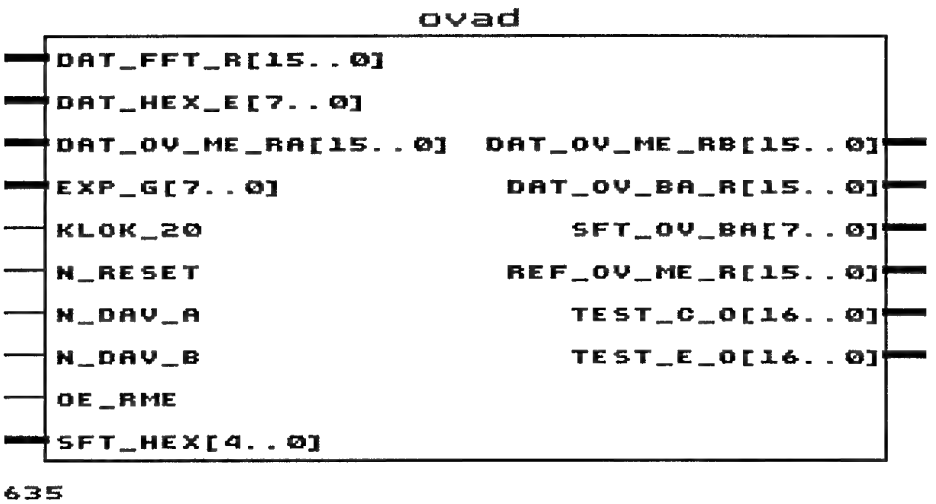
Figuur C.2: De FPGA die zorgt voor de besturing van de linkerkant van het DPRAM.



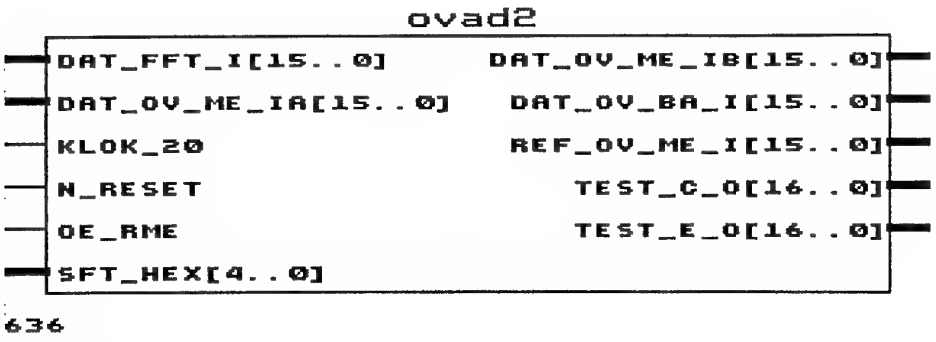
Figuur C.3: 2 FPGA's, schaling van de data.



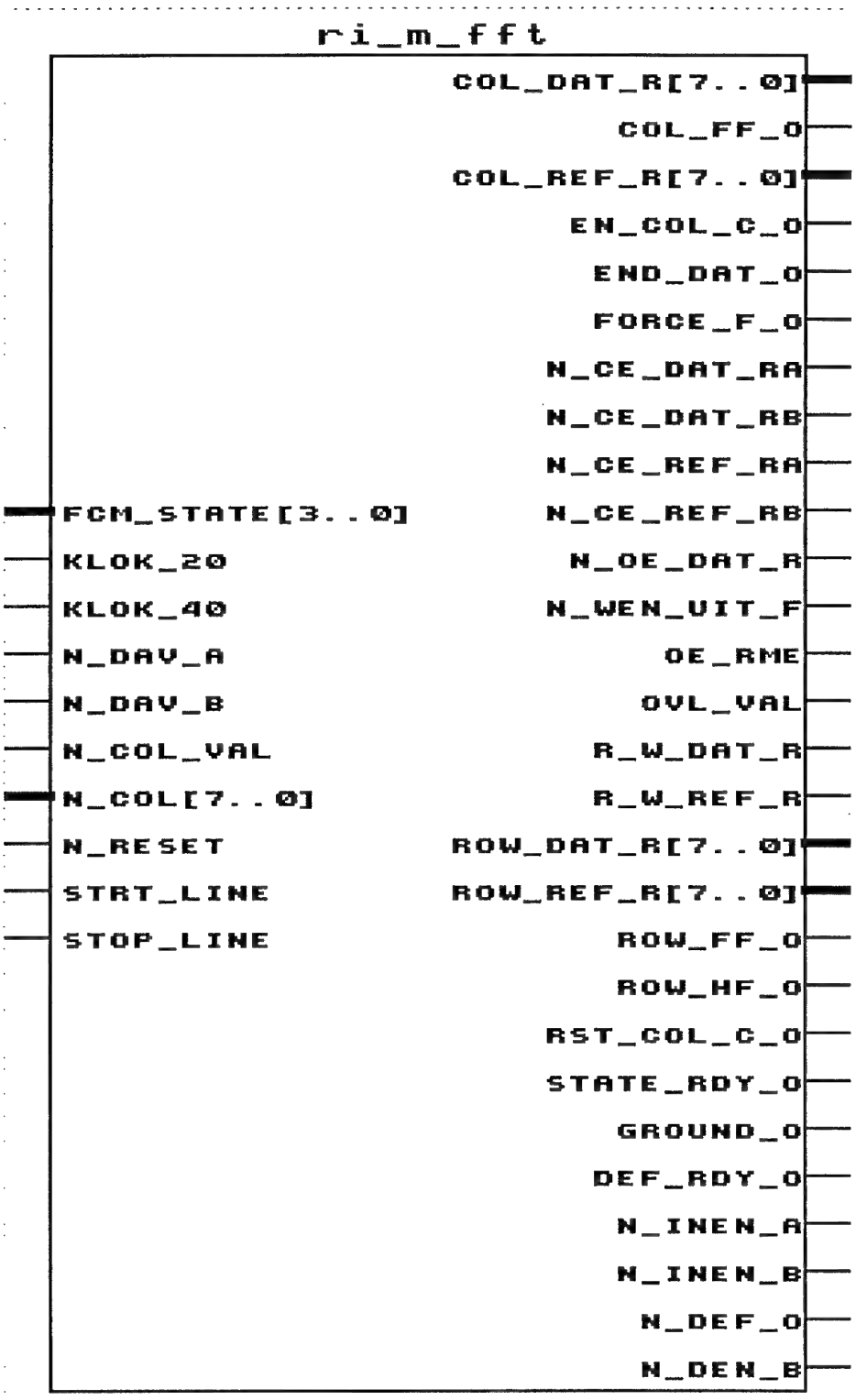
Figuur C.4: Bepaling van de nieuwe exponent.



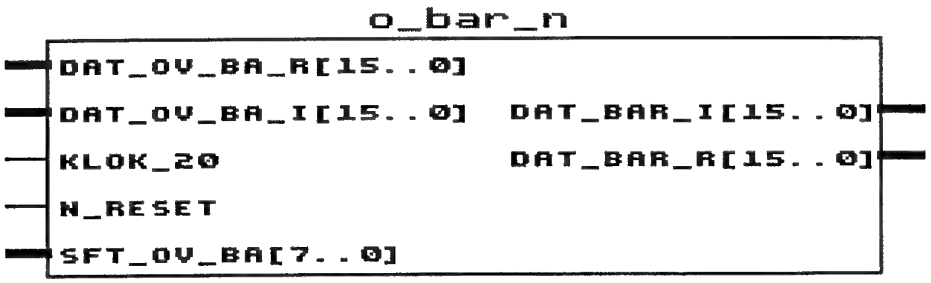
Figuur C.5: FPGA die zorgt voor het optellen van twee reële reeksen tijdens kolom IFFT.



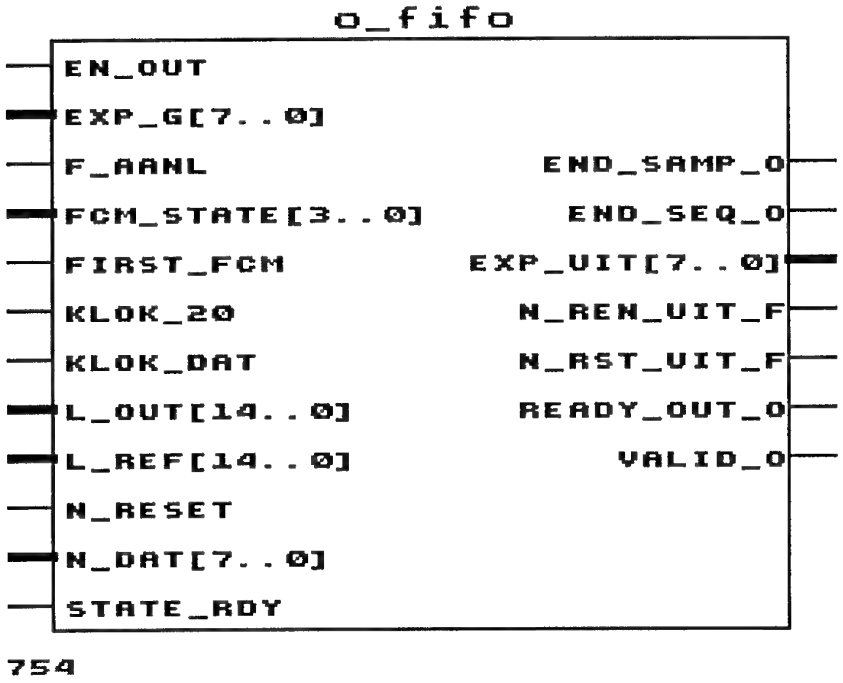
Figuur C.6: FPGA die zorgt voor het optellen van twee imaginaire reeksen tijdens kolom IFFT.



Figuur C.7: Besturing van de rechterkant van het DPRAM.



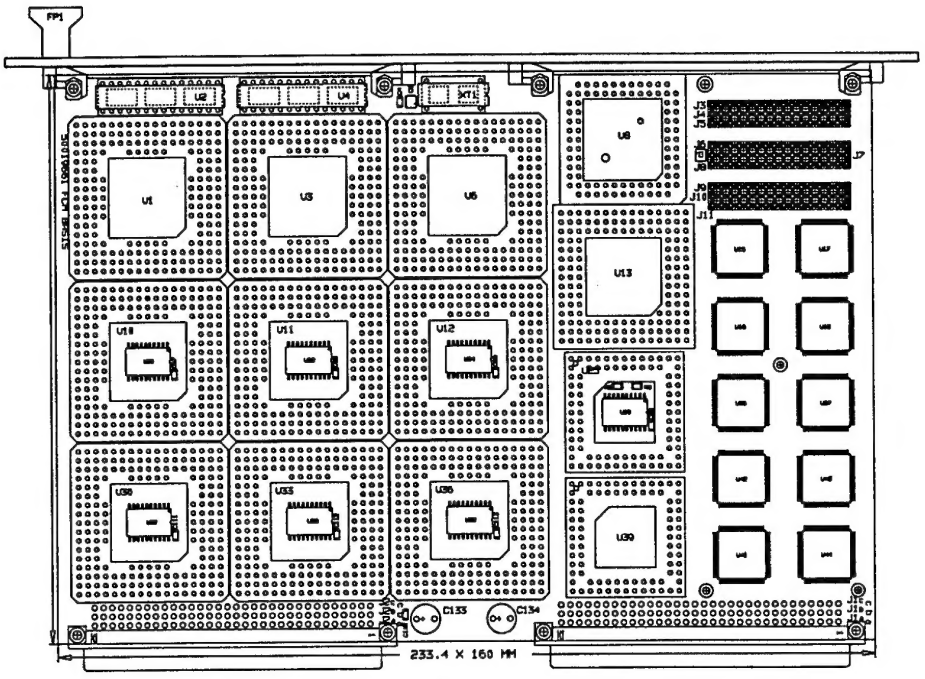
Figuur C.8: FPGA die zorgt voor de schaling van de uitgaande data.



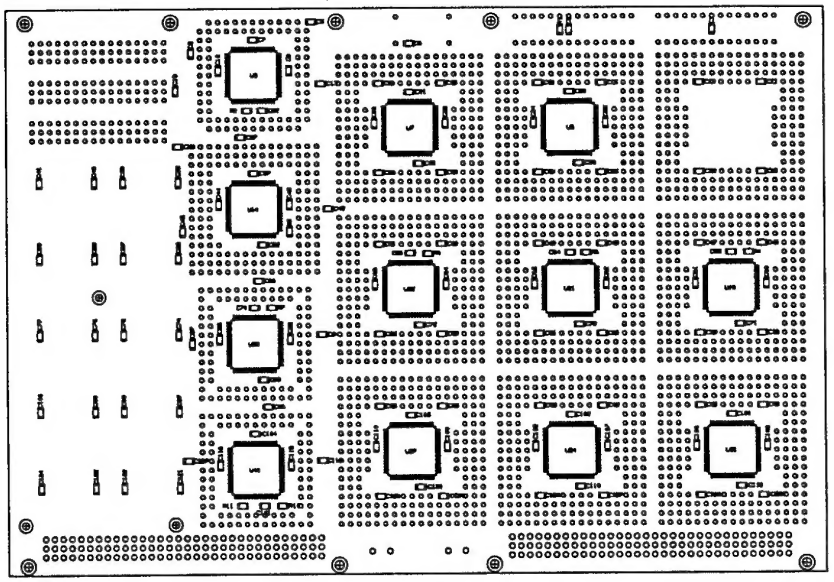
Figuur C.9: Besturing van de uitgangsfifo.

Bijlage D Layouts van de Printed Circuit Boards

De layout van de Printed Circuit Board (PCBs) is weergegeven in Figuur D.1. Wegens ruimtegebrek op de PCB zijn de geheugencomponenten op een opzetprint geplaats. Deze is weergegeven in Figuur D.2. De printkaart inclusief de opzetprint past in één U6 slot van een rack.

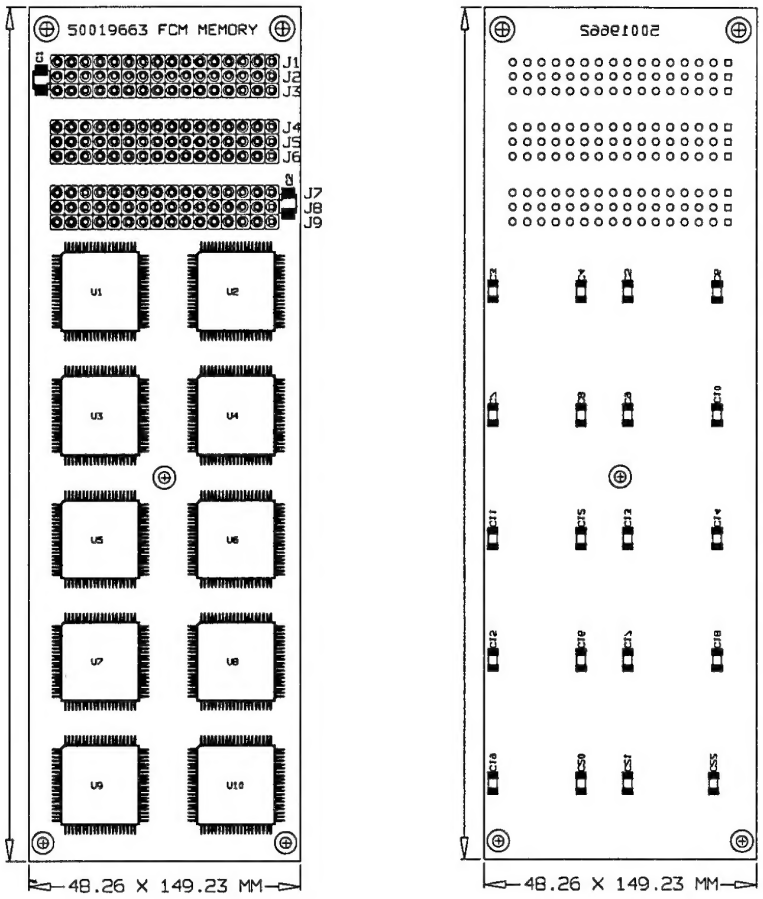


(a)



(b)

Figuur D.1: Bovenkant (a) en onderkant (b) van de printplaat.



(a) *(b)*

Figuur D.2: Bovenkant (a) en onderkant (b) van de opzetprint.

REPORT DOCUMENTATION PAGE
(MOD-NL)

1. DEFENCE REPORT NO (MOD-NL)	2. RECIPIENT'S ACCESSION NO	3. PERFORMING ORGANIZATION REPORT NO
TD96-0400		FEL-96-A307
4. PROJECT/TASK/WORK UNIT NO	**5. CONTRACT NO**	**6. REPORT DATE**
23964	A93KLu777	October 1997
7. NUMBER OF PAGES	**8. NUMBER OF REFERENCES**	**9. TYPE OF REPORT AND DATES COVERED**
67 (incl 4 appendices, excl RDP & distribution list)	3	

10. TITLE AND SUBTITLE

Fast Convolution Module
(Fast Convolution Module)

11. AUTHOR(S)

Dr L.H.J. Bierens, P.C.R. Beukelman, C. van 't Wout

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TNO Physics and Electronics Laboratory, PO Box 96864, 2509 JG The Hague, The Netherlands
Oude Waalsdorperweg 63, The Hague, The Netherlands

13. SPONSORING AGENCY NAME(S) AND ADDRESS(ES)

DMKLu/MXST1
Binckhorstlaan 135, 2516 BA The Hague, The Netherlands

14. SUPPLEMENTARY NOTES

The classification designation Ongerubriceerd is equivalent to Unclassified, Stg. Confidentieel is equivalent to Confidential and Stg. Geheim is equivalent to Secret.

15. ABSTRACT (MAXIMUM 200 WORDS (1044 BYTE))

This report describes the design and realisation of a real-time range/azimuth compression module, the so-called 'Fast Convolution Module', based on the fast convolution algorithm developed at TNO-FEL. The module will be part of a Real-Time SAR Processor for Air Force applications. Range and azimuth compression, as well as the Corner Turning Memory, are the critical steps within the SAR processing chain. The result of this project, a tested hardware module, shows the feasibility of the technological concept of the 'Fast Convolution Module'. The module is tested in the Real-Time SAR Testbed environment, that has been developed at TNO-FEL. The realisation of this module is an important step in the realisation of a demonstrator Real-Time SAR Processor.

16. DESCRIPTORS	IDENTIFIERS
Real-Time Computation	Digital Signal Processing
Synthetic Aperture Radar	Fast Fourier Transform
Data Processing	
Signal Processing	

17a. SECURITY CLASSIFICATION (OF REPORT)	17b. SECURITY CLASSIFICATION (OF PAGE)	17c. SECURITY CLASSIFICATION (OF ABSTRACT)
Ongerubriceerd	Ongerubriceerd	Ongerubriceerd

18. DISTRIBUTION AVAILABILITY STATEMENT	17d. SECURITY CLASSIFICATION (OF TITLES)
Unlimited Distribution	Ongerubriceerd

Distributielijst

1.	Bureau TNO Defensieonderzoek
2.	Directeur Wetenschappelijk Onderzoek en Ontwikkeling*)
3.	HWO-KL*)
4.	HWO-KLu
5.	HWO-KM*)
6.	HWO-CO*)
7 t/m 9.	KMA, Bibliotheek
10.	DMKLu/MXST1, t.a.v. Ir. S.J.J. de Bruin
11.	TU Delft, Vakgroep CAS, Dr. ir. E.F. Deprettere
12.	Directie TNO-FEL, t.a.v. Dr. J.W. Maas
13.	Directie TNO-FEL, t.a.v. Ir. J.A. Vogel, daarna reserve
14.	Archief TNO-FEL, in bruikleen aan M&P*)
15.	Archief TNO-FEL, in bruikleen aan Drs. Tj. de Groot*)
16.	Archief TNO-FEL, in bruikleen aan Dr. ir. J.L.J. de Sonneville
17.	Archief TNO-FEL, in bruikleen aan Ir. J.P. van Bezouwen
18.	Archief TNO-FEL, in bruikleen aan Ir. A.W.P. van Heijningen
19.	Archief TNO-FEL, in bruikleen aan Ing. J.C.P. Bol
20.	Archief TNO-FEL, in bruikleen aan Ing. P.A.M. Stijnman
21.	Archief TNO-FEL, in bruikleen aan Ir. P. Hoogeboom
22.	Archief TNO-FEL, in bruikleen aan Dr. ir. G.J. Rijckenberg
23.	Archief TNO-FEL, in bruikleen aan Dr. ir. L.H.J. Bierens
24.	Archief TNO-FEL, in bruikleen aan Ir. P.C.R. Beukelman
25.	Archief TNO-FEL, in bruikleen aan Ing. C. van 't Wout
26.	Documentatie TNO-FEL
27.	Reserve
	TNO-PML, Bibliotheek**)
	TNO-TM, Bibliotheek**)
	TNO-FEL, Bibliotheek**)

Indien binnen de krijgsmacht extra exemplaren van dit rapport worden gewenst door personen of instanties die niet op de verzendlijst voorkomen, dan dienen deze aangevraagd te worden bij het betreffende Hoofd Wetenschappelijk Onderzoek of, indien het een K-opdracht betreft, bij de Directeur Wetenschappelijk Onderzoek en Ontwikkeling.

*) Beperkt rapport (titelblad, managementuittreksel, RDP en distributielijst).
**) RDP.